Chapter 21

Vibration Analysis of Laminated Composite Variable Thickness Plate Using Finite Strip Transition Matrix Technique and MATLAB Verifications

Wael A. Al-Tabey

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/57384

1. Introduction

In the past, the model of thin plate on the elastic foundation was mainly used in structural applications. Currently, thin films of metal, ceramic or synthetic materials deposited on the surface of the structural parts of the electronic devices are used to improve their mechanical, thermal, electrical and tribological properties. These thin films of material are considered as thin plates and in these applications, the substrate of thin film can be simulated as an elastic foundation [1-2].

The laminated composite rectangular plate is very common in many engineering fields such as aerospace industries, civil engineering and marine engineering. The ability to conduct an accurate free vibration analysis of plates with variable thickness is absolutely essential if the designer is concerned with possible resonance between the plate and driving force [3].

Ungbhakorn and Singhatanadgid [4] investigated the buckling problem of rectangular laminated composite plates with various edge supports by using an extended Kantorovich method is employed.

Setoodeh, Karami [5] investigated A three-dimensional elasticity approach to develop a general free vibration and buckling analysis of composite plates with elastic restrained edges.

Luura and Gutierrez [6] studied the vibration of rectangular plates by a non-homogenous elastic foundation using the Rayleigh-Ritz method.

Ashour [7] investigated the vibration analysis of variable thickness plates in one direction with edges elastically restrained against both rotation and translation using the finite strip transition matrix technique.

Grossi, Nallim [8] investigated the free vibration of anisotropic plates of different geometrical shapes and generally restrained boundaries. An analytical formulation, based on the Ritz method and polynomial expressions as approximate functions for analyzing the free vibrations of laminated plates with smooth and non-smooth boundary with non classical edge supports is presented.

LU, et al [9] presented the exact analysis for free vibration of long-span continuous rectangular plates based on the classical Kirchhoff plate theory, using state space approach associated with joint coupling matrices.

Chopra [10] studied the free vibration of stepped plates by analytical method. Using the solutions to the differential equations for each region of the plate with uniform thickness, he formulated the overall Eigen value problem by introducing the boundary conditions and continuity conditions at the location of abrupt change of thickness. However this method suffers from the drawback of excessive continuity, as in theory the second and third derivatives of the deflection function at the locations of abrupt change of thickness should not be continuous.

Cortinez and Laura [11] computed the natural frequencies of stepped rectangular plates by means of the Kantorovich extended method, whereby the accuracy was improved by inclusion of an exponential optimization parameter in the formulation.

Bambill et al. [12] subsequently obtained the fundamental frequencies of simply supported stepped rectangular plates by the Rayleigh–Ritz method using a truncated double Fourier expansion.

Laura and Gutierrez [13] studied the free vibration problem of uniform rectangular plates supported on a non-homogeneous elastic foundation based on the Rayleigh–Ritz method using polynomial coordinate functions which identically satisfy the governing boundary conditions.

Harik and Andrade [14] used the "analytical strip method" to the stability analysis of unidirectionally stepped plates. In essence, the stepped plate is divided into rectangular regions of uniform thickness. The differential equations of stability for each region are solved and the continuity conditions at the junction lines as well as the boundary conditions are then imposed.

1.1. The chapter aims

This chapter presents the finite strip transition matrix technique (FSTM) and a semi-analytical method to obtain the natural frequencies and mode shapes of symmetric angle-ply laminated composite rectangular plate with classical boundary conditions (S-S-F-F). The plate has a uniform thickness in x direction and varying thickness h(y) in y direction, as shown in Figure 1. The boundary conditions in the variable thickness direction are simply supported and they are satisfied identically and the boundary conditions in the other direction are free and are approximated. Numerical results for simple-free (S-S-F-F) boundary conditions at the plate edges are presented. The illustrated results are in excellent agreement compared with solutions available in the literature, which validates the accuracy and reliability of the proposed technique.

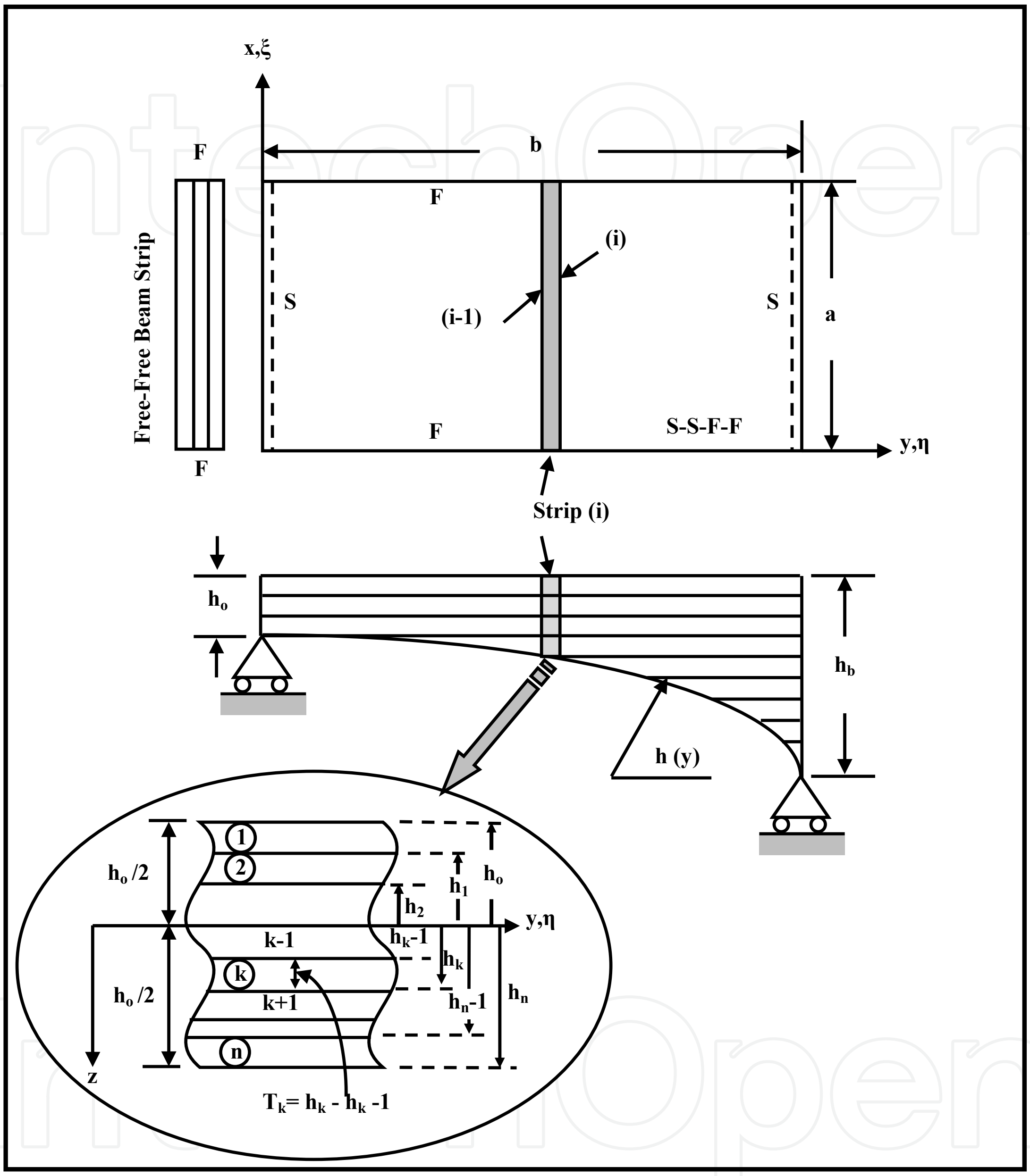

Figure 1. A rectangular laminated plate with variable thickness

2. Formulation

The equation of motion governing the vibration of rectangular plate under the assumption of the classical deformation theory in terms of the plate deflection W(x, y, t) is given by:

$$\frac{\partial^2 M_X}{\partial x^2} - 2\frac{\partial^2 M_{XY}}{\partial x \partial y} + \frac{\partial^2 M_Y}{\partial y^2} = -\rho \; h(y)\frac{\partial^2 w_o}{\partial t^2} \tag{1}$$

Where W is the transverse deflection, ϱ = the density per unit area of the plate and h(y) is the plate thickness at any point. The bending and the twisting moments in terms of displacements are given by:

$$\left.\begin{aligned} M_X &= -D_{11}\frac{\partial^2 w_o}{\partial x^2} - D_{12}\frac{\partial^2 w_o}{\partial y^2} - 2D_{16}\frac{\partial^2 w_o}{\partial x \partial y} \\ M_Y &= -D_{12}\frac{\partial^2 w_o}{\partial x^2} - D_{22}\frac{\partial^2 w_o}{\partial y^2} - 2D_{26}\frac{\partial^2 w_o}{\partial x \partial y} \\ M_{XY} &= -D_{16}\frac{\partial^2 w_o}{\partial x^2} - D_{26}\frac{\partial^2 w_o}{\partial y^2} - 2D_{66}\frac{\partial^2 w_o}{\partial x \partial y} \end{aligned}\right\} \tag{2}$$

The flexural rigidities D_{ij} of the plate are given by:

$$D_{ij} = \frac{1}{3}\frac{h^3(y)}{h_o^3}\sum_{k=1}^{n}\left[(\bar{Q}_{ij})\right]_k (h_{ok}^3 - h_{ok-1}^3), \qquad i,j = 1,2,3,......... \tag{3}$$

Where h_{ok} is the distance from the middle-plane of the plate according to h_o to the bottom of the h_{oth} layer as shown in Figure 1. And $\overline{Q_{ij}^k}$ are the plane stress transformed reduced stiffness coefficients of the lamina in the laminate Cartesian coordinate system. They are related to reduced stiffness coefficients of the lamina in the material axes of lamina Q_{ij}^k by proper coordinate relationships they can be expressed in terms of the engineering notations as:

$$Q_{ij} = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{12} & Q_{22} & Q_{23} \\ Q_{13} & Q_{23} & Q_{66} \end{bmatrix} = \begin{bmatrix} \frac{E_{11}}{(1-\upsilon_{12}\upsilon_{21})} & \frac{\upsilon_{21}E_{11}}{(1-\upsilon_{12}\upsilon_{21})} & 0 \\ \frac{\upsilon_{21}E_{11}}{(1-\upsilon_{12}\upsilon_{21})} & \frac{E_{22}}{(1-\upsilon_{21}\upsilon_{12})} & 0 \\ 0 & 0 & G_{12} \end{bmatrix} \tag{4}$$

Where E_{11}, E_{22} are the longitudinal and transverse young's moduli parallel and perpendicular to the fiber orientation, respectively and G_{12} is the plane shear modulus of elasticity, υ_{12} and υ_{21} are the poisson's ratios. Thus, the governing partial differential equation of laminated composite rectangular plate with variable thickness as shown in Figure 1 is reduced to:

http://dx.doi.org/10.5772/57384

$$D_{11}\frac{\partial^4 w_o}{\partial x^4}+4D_{16}\frac{\partial^4 w_o}{\partial x^3\partial y}+2(D_{12}+2D_{66})\frac{\partial^4 w_o}{\partial x^2\partial y^2}+4D_{26}\frac{\partial^4 w_o}{\partial x\partial y^3}+D_{22}\frac{\partial^4 w_o}{\partial y^4}=-\rho\ h(y)\frac{\partial^2 w_o}{\partial t^2} \tag{5}$$

Or in contraction form:

$$D_{11}W_{xxxx}+4D_{16}W_{xxxy}+2(D_{12}+2D_{66})W_{xxyy}+4D_{26}W_{xyyy}+D_{22}W_{yyyy}=-\rho\ h(y)W_{tt} \tag{6}$$

The substitution of equation (3) into equation (6) given the governing Partial differential equation:

$$\begin{aligned}&D_{11}\left\{\frac{\partial^2}{\partial x^2}\left(\frac{h^3(y)}{h_o^3}W_{,xx}\right)\right\}+2(D_{12}+2D_{66})\left\{\frac{\partial^2}{\partial x\partial y}\left(\frac{h^3(y)}{h_o^3}W_{,xy}\right)\right\}+D_{16}\left\{\frac{\partial^2}{\partial x^2}\left(\frac{h^3(y)}{h_o^3}W_{,xy}\right)\right\}\\&+4D_{26}\left\{\frac{\partial^2}{\partial y^2}\left(\frac{h^3(y)}{h_o^3}W_{,xy}\right)\right\}+D_{22}\left\{\frac{\partial^2}{\partial y^2}\left(\frac{h^3(y)}{h_o^3}W_{,yy}\right)\right\}=-m_o\frac{h(y)}{h_o}W_{tt}\end{aligned} \tag{7}$$

Equation (7) may be written as:

$$\begin{aligned}&D_{11}\frac{h^3(y)}{h_o^3}W_{xxxx}+\left(\frac{2(D_{12}+2D_{66})}{h_o^3}\right)\frac{\partial h^3(y)}{\partial y}W_{xxy}+\left(\frac{2(D_{12}+2D_{66})}{h_o^3}\right)h^3(y)W_{xxyy}+D_{16}\frac{h^3(y)}{h_o^3}W_{xxxy}\\&+\left(\frac{4D_{26}}{h_o^3}\frac{\partial^2 h^3(y)}{\partial y^2}\right)W_{xy}+\frac{4D_{26}}{h_o^3}h^3(y)W_{xyyy}+\frac{8D_{26}}{h_o^3}\frac{\partial h^3(y)}{\partial y}W_{xyy}+\left(\frac{D_{22}}{h_o^3}\frac{\partial^2 h^3(y)}{\partial y^2}\right)W_{yy}\\&+\frac{D_{22}}{h_o^3}h^3(y)W_{yyyy}+\frac{2D_{22}}{h_o^3}\frac{\partial h^3(y)}{\partial y}W_{yyy}=-m_o\frac{h(y)}{h_o}W_{tt}\end{aligned} \tag{8}$$

The equation of motion (8) can be normalized using the non-Dimensional variables ξ and η as follows :

$$\begin{aligned}&\psi_1\frac{1}{a^4}W_{\xi\xi\xi\xi}+\frac{2\psi_2}{h^3(\eta)}\frac{1}{a^2b}\frac{\partial h^3(\eta)}{\partial\eta}W_{\xi\xi\eta}+2\psi_2\frac{1}{a^2b^2}W_{\xi\xi\eta\eta}+\psi_3\frac{1}{a^3b}W_{\xi\xi\xi\eta}+4\psi_4\frac{1}{ab^3}W_{\xi\eta\eta\eta}\\&+\frac{1}{ab}\frac{4\psi_4}{h^3(\eta)}\frac{\partial^2 h^3(\eta)}{\partial\eta^2}W_{\xi\eta}+\frac{8\psi_4}{h^3(\eta)}\frac{1}{ab^2}\frac{\partial h^3(\eta)}{\partial\eta}W_{\xi\eta\eta}+\frac{1}{b^2}\frac{1}{h^3(\eta)}\frac{\partial^2 h^3(\eta)}{\partial\eta^2}W_{\eta\eta}\\&+\frac{1}{b^4}W_{\eta\eta\eta\eta}+\frac{2}{h^3(\eta)}\frac{1}{b^3}\frac{\partial h^3(\eta)}{\partial\eta}W_{\eta\eta\eta}=-\frac{m_o}{D_{22}}\frac{h_o^2}{h^2(\eta)}W_{tt}\end{aligned} \tag{9}$$

Where $\beta=\frac{a}{b}$ is the aspect ratio, $\xi=\frac{x}{a}$, $\eta=\frac{y}{b}$, $\psi_1=\frac{D_{11}}{D_{22}}$, $\psi_2=\frac{(D_{12}+2D_{66})}{D_{22}}$, $\psi_3=\frac{D_{16}}{D_{22}}$ and $\psi_4=\frac{D_{26}}{D_{22}}$.

3. Method of solution

The displacement $W(\xi, \eta, t) = W(\xi, \eta)e^{i\omega t}$ can be expressed in terms of the shape function $X_i(\xi)$, chosen a prior; and the unknown function $Y_i(\eta)$as:

$$W(\xi,\eta,t) = \sum_{i=0}^{N} X_i(\xi)Y_i(\eta)\ e^{i\omega t} \tag{10}$$

The most commonly used is the Eigen function obtained from the solution of beam free vibration under the prescribed boundary conditions at ξ=0 and ξ=1.

The free vibration of a beam of length a can be described by the non-Dimensional differential equation:

$$\left(\frac{2F_{T1}\mu_i\left(F_{R1}F_{R2} - \mu_i^2\right)}{\left(\mu_i^4 + F_{R1}F_{T1}\right)} \sin\mu_i + \frac{2F_{T1}\mu_i\left(F_{R1} + F_{R2}\right)}{\left(\mu_i^4 + F_{R1}F_{T1}\right)} \mu_i \cos\mu_i \right) A_3 \tag{11}$$

Where EI is the flexural rigidity of the beam. The boundary conditions for free edges beam as shown in Fig. 2 are:

at ξ=0 and ξ=1

$$\left.\begin{aligned} \frac{\partial^2 X_i(\xi)}{\partial \xi^2} = 0 \\ \frac{\partial^3 X_i(\xi)}{\partial \xi^3} = 0 \end{aligned}\right\} \tag{12}$$

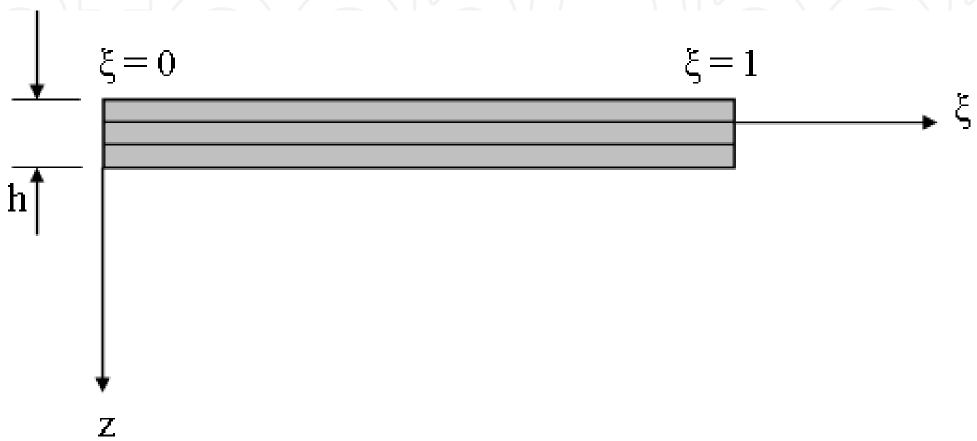

Figure 2. The two free edges beam strip in ξ-direction

In this paper, the beam shape function in ξ-direction is considered as a strip element of the plate and the flexural rigidity EI of the beam can replaced by $(1-\upsilon^2)D_{22}$ and for $\upsilon=0.3$, it can be just approximated by $E \approx D_{22}$. The solution of the beam equation is given as:

$$X_i(\xi) = A_1 \sin(\mu_i\xi) + A_2 \cos(\mu_i\xi) + A_3 \sinh(\mu_i\xi) + A_4 \cosh(\mu_i\xi) \tag{13}$$

One can obtain the following system of homogenous linear equations by satisfying the boundary conditions (12) at ξ=0 and ξ=1.

$$\left.\begin{aligned} \Phi_i &= \frac{-\sinh\mu_i + \sin\mu_i}{\cosh\mu_i - \cos\mu_i} \\ X_i(\xi) &= \sin(\mu_i\xi) - \sinh(\mu_i\xi) + \frac{1}{\Phi_i}\left(\cos(\mu_i\xi) - \cosh(\mu_i\xi)\right) \end{aligned}\right\} \tag{14}$$

The different value of μ_i are the roots of equation:

$$\frac{-2\cos(\mu_i)\cosh(\mu_i) + \cos^2(\mu_i) + \sin^2(\mu_i) - \sinh^2(\mu_i) + \cosh^2(\mu_i)}{\sinh(\mu_i)\cosh(\mu_i) + \sin(\mu_i)\cosh(\mu_i) - \sinh(\mu_i)\cos(\mu_i) - \sin(\mu_i)\cos(\mu_i)} = 0 \tag{15}$$

The roots of equation (15) are represented in the recurrence form:

$$\mu_i = (i + 0.5)\pi, \qquad i = 0,\ 1,\ 2,\ 3,\ .... \tag{16}$$

The substitution of equation (10) into equation (9), multiplying both sides by $X_j(x)$ and after some manipulation, we can find:

$$\begin{aligned} &\sum_{i=0}^{N}\sum_{j=0}^{M} \frac{\beta^4}{f_3(\eta)} Y_{i,\eta\eta\eta\eta} + 2\beta^3 a \frac{f_1(\eta)}{f_3(\eta)} Y_{i,\eta\eta\eta} + \Big(\frac{2\psi_2\beta^2}{f_3(\eta)}\frac{c_{ij}}{a_{ij}} + 8\psi_4\beta^2 a \frac{f_1(\eta)}{f_3(\eta)}\frac{b_{ij}}{a_{ij}} + \beta^2 a^2 \frac{f_2(\eta)}{f_3(\eta)}\Big) Y_{i,\eta\eta} \\ &+\Big(2\psi_2\beta a \frac{f_1(\eta)}{f_3(\eta)}\frac{c_{ij}}{a_{ij}} + \frac{\psi_3\beta}{f_3(\eta)}\frac{d_{ij}}{a_{ij}} + 4\psi_4\beta a^2 \frac{f_2(\eta)}{f_3(\eta)}\frac{b_{ij}}{a_{ij}} + \frac{4\psi_4\beta^3}{f_3(\eta)}\frac{b_{ij}}{a_{ij}}\Big) Y_{i,\eta} \\ &+\Big(\frac{\psi_1}{f_3(\eta)}\frac{e_{ij}}{a_{ij}} - \lambda^2\Big) Y_i = 0 \end{aligned} \tag{17}$$

Where $\lambda^2 = \dfrac{m_o\omega^2 a^4}{D_{22}}$, $f_1(\eta) = \dfrac{1}{h^3(\eta)}\dfrac{\partial h^3(\eta)}{\partial\eta}$, $f_2(\eta) = \dfrac{1}{h^3(\eta)}\dfrac{\partial^2 h^3(\eta)}{\partial\eta^2}$, $f_3(\eta) = \dfrac{h_o^2}{h^2(\eta)}$,

$$a_{ij}=\int_0^1 X_i X_j d\xi,$$

$$b_{ij}=\int_0^1 X_j X_{i,\xi} d\xi,$$

$$c_{ij}=\int_0^1 X_j X_{i,\xi\xi} d\xi,$$

$$d_{ij}=\int_0^1 X_j X_{i,\xi\xi\xi} d\xi$$

and $e_{ij}=\int_0^1 X_j X_{i,\xi\xi\xi\xi} d\xi.$

From the orthogonality of the beam Eigen function, $a_{ij}=e_{ij}=0$ for $i\neq j$, this is true for all boundary conditions except for plates having free edges in the ξ-direction.

The system of fourth order partial differential equations in equation (17) can be reduced to a system of first order homogeneous ordinary differential equations:

$$\frac{d}{d\eta}\{Y_k\}_{ij}=[A_i]_k\{Y_k\}_{ij} \tag{18}$$

And after some manipulation, the governing differential equation (17) will become:

$$\sum_{i=0}^{N}\sum_{j=0}^{M} E_{ij}Y_i^{////}+\frac{(O_1)_{ij}}{(O_0)_{ij}}Y_i^{///}+\frac{(O_2)_{ij}}{(O_0)_{ij}}Y_i^{//}+\frac{(O_3)_{ij}}{(O_0)_{ij}}Y_i^{/}+\frac{(O_4)_{ij}-\lambda^2}{(O_0)_{ij}}Y_i=0 \tag{19}$$

Where the frame denotes differentiation with respect to η.

Where: $(O_0)_{ij}=\beta^4 t_1(\eta)E_{ij}$, $(O_1)_{ij}=2\beta^3 a t_2(\eta)E_{ij}$, $(O_2)_{ij}=(2\psi_2\beta^2 t_1(\eta)\frac{c_{ij}}{a_{ij}}+8\psi_4\beta^2 a t_2(\eta)\frac{b_{ij}}{a_{ij}}+\beta^2 a^2 t_3(\eta))$

$(O_3)_{ij}=(2\psi_2\beta a t_2(\eta)\frac{c_{ij}}{a_{ij}}+\psi_3\beta\, t_1(\eta)\frac{d_{ij}}{a_{ij}}+4\psi_4\beta a^2 t_3(\eta)\frac{b_{ij}}{a_{ij}}+4\psi_4\beta^3 t_1(\eta)\frac{b_{ij}}{a_{ij}})$, $(O_4)_{ij}=\psi_1 t_1(\eta)\frac{e_{ij}}{a_{ij}}$,

$[E_{ij}]=i\times j$ Unit matrix,

i= 0, 1, 2, 3, ……….,N, j= 0, 1, 2, 3, ……….,M where the coefficients of the matrix $[A_i]_k$ in equation (18), in general, are functions of η and the Eigen value parameter λ. The vector Y_k is given by:

$$Y_k = \begin{bmatrix} \overline{Y}_1 & \overline{Y}_2 & \text{K} & \overline{Y}_i & \text{K} & \overline{Y}_N \end{bmatrix} \tag{20}$$

Where:

$$\overline{Y}_i = \begin{bmatrix} Y_i & Y_i^{/} & Y_i^{//} & Y_i^{///} \end{bmatrix} \tag{21}$$

Solving the above system of first order ordinary differential equations using the transition matrix technique yields, at any strip element (i) with boundaries (i-1) and (i) to,

$$\{Y_i\}_j = [B_i]_j \{Y_{i-1}\}_j \tag{22}$$

Where $[B_i]_j$ is called the transition matrix of the strip element (i), which can be obtained using the method of system linear differential equations of the strip element (i) in equation (18) (the exact solution of (ODE)).

Following the same procedure, the above boundary conditions (equations (12)) can be written. The simple boundary conditions at η=0 and η=1 as shown in Figure 3 are:

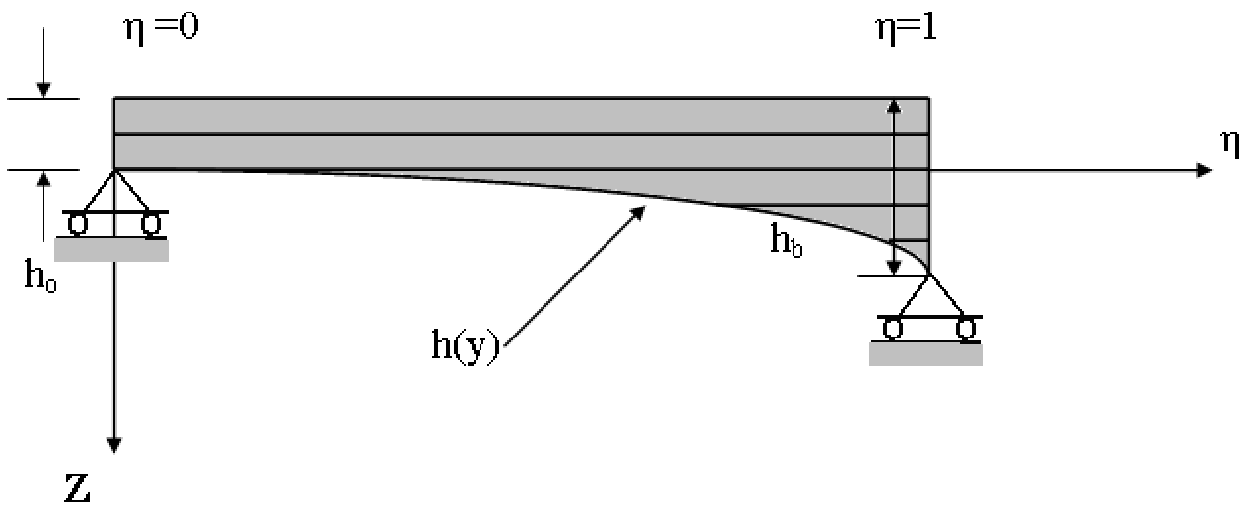

Figure 3. The two edges clamped variable thickness beam strip in η-direction

The boundary conditions at η=0 and η=1 can be expressed as:

$$\left.\begin{array}{c} w_o = 0 \\ -\frac{D_{12}}{a^2}\frac{\partial^2 w_o}{\partial \xi^2} - \frac{D_{22}}{b^2}\frac{\partial^2 w_o}{\partial \eta^2} - \frac{2D_{26}}{ab}\frac{\partial^2 w_o}{\partial \xi \partial \eta} = 0 \end{array}\right\} \tag{23}$$

Using the assumed solution, equation (10) the boundary conditions can be given by the following equations:

At η=0 and η=1

$$\left.\begin{array}{c} Y_i = 0 \\ \frac{\partial^2 Y_i}{\partial \eta^2} = \sum_{i=0}^{N} -\frac{2\psi_4 b_{ij}}{\beta a_{ij}}\frac{\partial Y_i}{\partial \eta} - \frac{\psi_5 c_{ij}}{\beta^2 a_{ij}} Y_i \end{array}\right\} \tag{24}$$

Or in contraction form:

$$\left.\begin{array}{c} Y_i = 0 \\ Y_i^{//} = \sum_{i=0}^{N} -CF_1\frac{b_{ij}}{a_{ij}} Y_i^{/} - CF_2\frac{c_{ij}}{a_{ij}} Y_i \end{array}\right\} \tag{25}$$

Where $CF_1 = \frac{2\psi_4}{\beta}$, $CF_2 = \frac{\psi_5}{\beta^2}$, $\psi_5 = \frac{D_{12}}{D_{22}}$

The solution is found using 2N initial vectors Y_0 at η=0. Equation (22) is applied across the stripped plate until the final end at η=1 is reached. Thus, 2N solutions S_i, where i= 0, 1, 2, 3, ……….,N, can be obtained. The true solutions $[S]$ can be written as a linear combination of these solutions [7]:

$$[S] = \sum_{i=1}^{2N} C_i S_i \tag{26}$$

Where C_i are arbitrary constants. These constants can be determined by satisfying 2N boundary conditions at η=1 [7]. The matrix $[S]$ forms a standard Eigen value problem.

4. Numerical results and discussion

In this section, some numerical results are presented for symmetrically laminated, angle-ply variable thickness rectangular plate with simple support in the variable thickness direction

http://dx.doi.org/10.5772/57384

and free in the other direction. The designation (S-S-F-F) means that the edges x=0, x=a, y=0, y=b are free, free, simple supported and simple supported respectively. The plates are made up of five laminates with the fiber orientations [θ, - θ, θ, - θ, θ] and the composite material is Graphite/Epoxy, of which mechanical properties are given in Table 1. The Eigen frequencies obtained are expressed in terms of non-dimensional frequency parameter $\lambda = (\rho h_o \omega^2 a^4 / D_{22})^{1/2}$. To illustrate the solution, a plate with linear variable thickness, $h(y)$is used (see Appendix A).

$$h(\eta) = 1 + \Delta\eta \tag{27}$$

Where Δ is the tapered ratio of plate given by $\Delta = (h_b - h_0) / h_0$, (h_0) is the thickness of the plate at η=0 and (h_b) is the thickness of the plate at η=1. A convergence investigation is carried out for a uniform plate and for plate of variable thickness ($\Delta = 0.5$) with aspect ratio $\beta = (0.5,\ 1.0)$. By varying the harmonic numbers of the series solution in equation (10). The results are shown in Table 2. It is found that excellent agreement and stable and fast convergence can be achieved with only a few terms of series solution (N= 3 to 5).

Material	E1, (GPa)	E2, (GPa)	G12, (GPa)	υ12	E2/ E1	G12/ E1
Graphite/Epoxy	138	8.96	7.1	0.3	25	0.8

Table 1. Material properties of unidirectional composite

In order to validate the proposed technique, a comparison of the results with some results available for other numerical methods [15] for uniform laminated plates with simple support in the y-direction and free in the other direction. The first six natural frequencies of such uniform laminated plates are depicted in Table 2.

			Δ = 0.0			
N	**λ1**	**λ2**	**λ3**	**λ4**	**λ5**	**λ6**
1	70.4212	70.7012	140.4421	173.5211	180.6231	235.6753
2	70.4212	70.7012	140.4421	173.5211	180.6231	235.6753
3	70.2882	70.5827	140.2496	173.2098	180.2833	235.3197
4	70.2882	70.5827	140.2496	173.2098	180.2833	235.3197
5	70.2882	70.5827	140.2496	173.2098	180.2833	235.3197
Ref*	70.302	70.604	140.255	173.218	180.287	235.322

*Y.K. Cheung and D. Zhou [15].

Table 2. Comparison of the first six natural frequencies of symmetric angle-ply uniform laminated square plates ($\theta = 45°$), ($\beta = 1.0$)

Table 3 and Table 4 shows a convergence analysis of the first six frequencies parameters of symmetrically angle-ply five laminates [45/-45/45/-45/45] variable thickness plate with tapered ratio ($\Delta = 0.5$) and with aspect ratio $\beta = (0.5,\ 1.0)$ with simple support in the y-direction and free in the other direction (S-S-F-F).

Figure 4 and Figure 5 show the mode shapes of the first six fundamental frequencies of the above plate. Figure 4 and Figure 5 both are divided into two graphics. The first one shows the mode shapes of the plate in surface form and the other shows the mode shapes of the plate in surface contour form. All simulation results and graphics were obtained using MATLAB software.

Δ = 0.5						
β = 0.5						
N	**λ1**	**λ2**	**λ3**	**λ4**	**λ5**	**λ6**
1	80.2177	82.5621	155.9665	188.6633	194.6253	251.7333
2	80.2177	82.5621	155.9665	188.6633	194.6253	251.7333
3	79.8625	82.0025	155.3232	188.1111	194.1002	251.2035
4	79.8625	82.0025	155.3232	188.1111	194.1002	251.2035
5	79.8625	82.0025	155.3232	188.1111	194.1002	251.2035

Table 3. The first six frequencies parameter of S-S-F-F symmetrically angle-ply laminated [45/-45/45/-45/45] variable thickness plate ($\Delta = 0.5$), ($\beta = 0.5$).

Δ = 0.5						
β = 1.0						
N	**λ1**	**λ2**	**λ3**	**λ4**	**λ5**	**λ6**
1	72.7575	73.8666	143.3334	175.4963	183.7825	240.7621
2	72.7575	73.8666	143.3334	175.4963	183.7825	240.7621
3	72.1199	73.4444	142.9019	175.0024	183.1121	240.0159
4	72.1199	73.4444	142.9019	175.0024	183.1121	240.0159
5	72.1199	73.4444	142.9019	175.0024	183.1121	240.0159

Table 4. The first six frequencies parameter of S-S-F-F symmetrically angle-ply laminated [45/-45/45/-45/45] variable thickness plate ($\Delta = 0.5$), ($\beta = 1.0$)

http://dx.doi.org/10.5772/57384

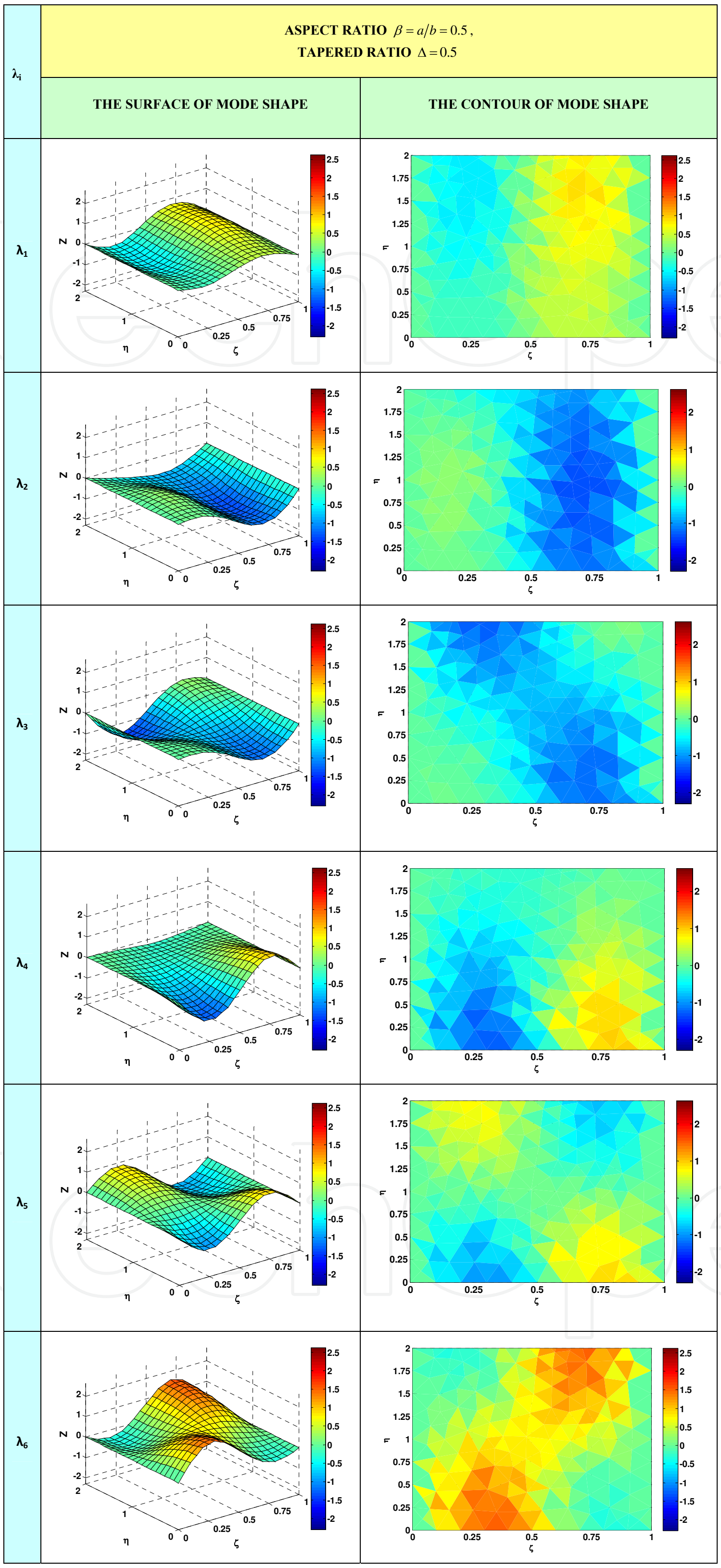

Figure 4. The mode shapes of the first six fundamental frequencies of the angle-ply symmetrically [45/-45/45/-45/45] laminated variable thickness rectangular plate with S-S-F-F edges, aspect ratio$\beta = a / b = 0.5$, tapered ratio $\Delta = 0.5$

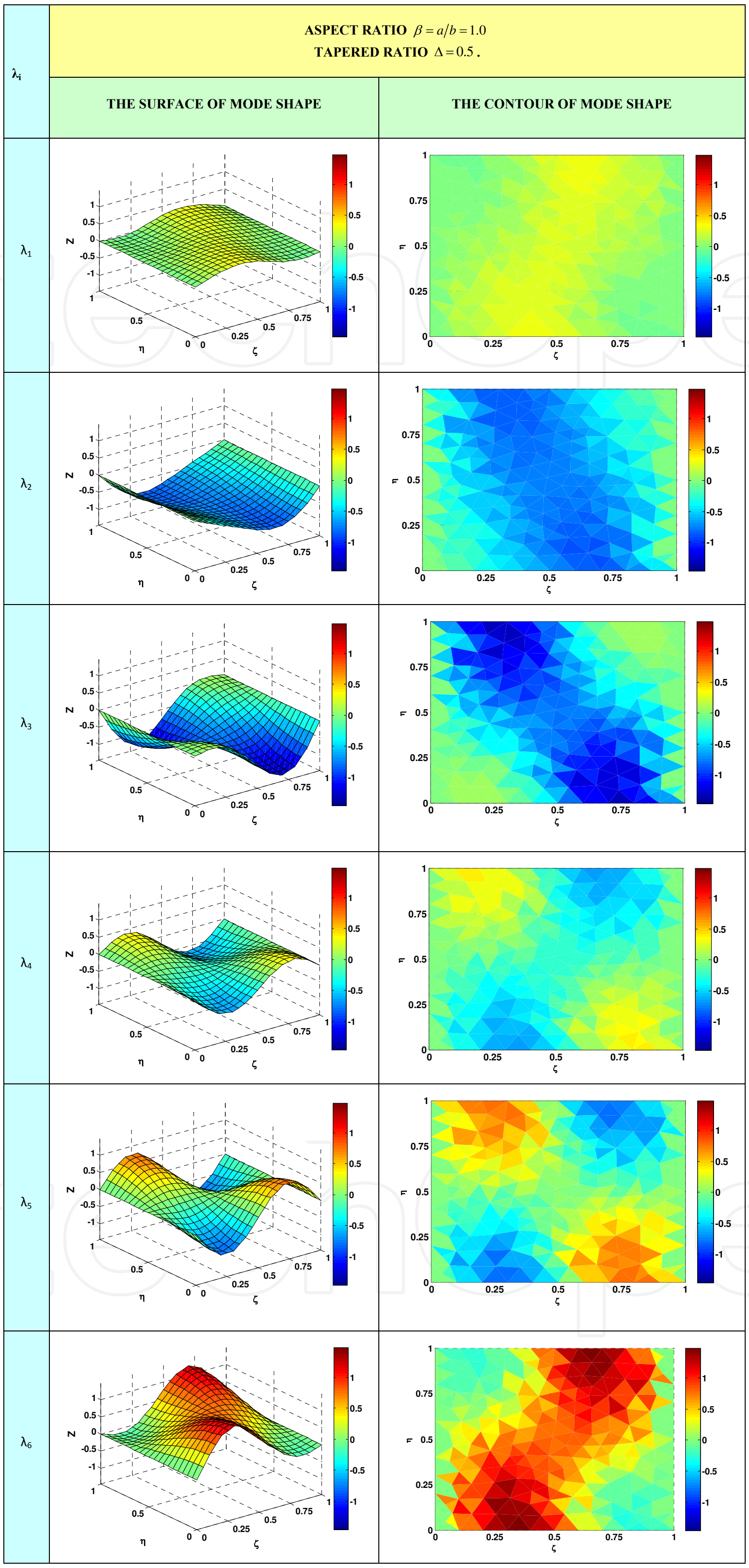

Figure 5. The mode shapes of the first six fundamental frequencies of the angle-ply symmetrically [45/-45/45/-45/45] laminated variable thickness rectangular plate with S-S-F-F edges, aspect ratio $\beta = a / b = 1.0$, tapered ratio $\Delta = 0.5$.

http://dx.doi.org/10.5772/57384

5. Concluding remarks

A semi-analytical solution of the free vibration of angle-ply symmetrically laminated variable thickness rectangular plate with classical boundary condition (S-S-F-F) is investigated using the finite strip transition matrix technique (FSTM). The numerical results for uniform angle-ply symmetrically square plate with classical boundary condition (S-S-F-F) is presented and compared with some available results. The results agree very closely with other results available in the literature. It can be observed from Tables 2 and 3 that rapid convergence is achieved with small numbers of N in the series solution. Comparing to other techniques, the finite strip transition matrix (FSTM) proves to be valid enough in this kind of application. In all cases the FSTM method is easily implemented in a computer program a yields a fast convergence and reliable results. Also, the effect of the tapered ratio (Δ) and aspect ratio (β) on the fundamental natural frequencies and the mode shapes for five layers angle-ply symmetrically laminated variable thickness plates has been investigated for two cases of tapered ratio (uniform and variable thickness) and two cases of aspect ratio (square and rectangular). In fact the varying of the thickness and the increase the length (b) about a length (a) tend to increase the natural frequencies and the mode shapes of the laminated plate. The results from this investigation have been illustrated in the three dimensional surface contours for two different aspect ratios.

Appendix (A)

Plate thickness function

In this appendix the derivation of the relation of the plate thickness h(y) in y-direction as shown in the Figure 6 is given.

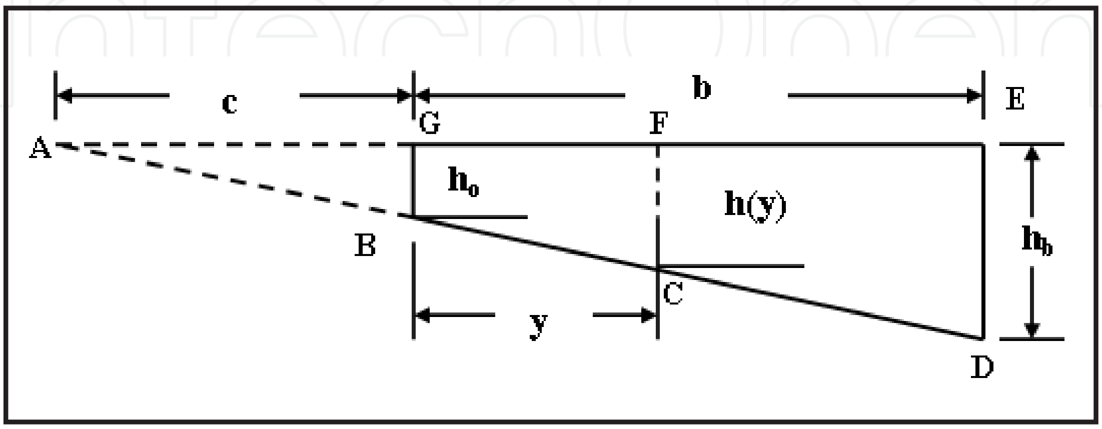

Figure 6. The relation of the plate thickness h(y) in y-direction

By similarity between the triangles (ABG) and (ACF):

$$h(y) = h_o(1 + \frac{y}{c}) \tag{28}$$

By similarity between the triangles (ABG) and (ADE):

$$\frac{h_o}{c} = \frac{h_b}{c+b} \tag{29}$$

From equations (28) and (29) the plate thickness relation is:

$$h(y) = h_o + \frac{(h_b - h_o)}{b} y \tag{30}$$

Where $h\,(y)=h_o$ at $y=0$,

$h\,(y)=h_b$ at $y=b$,

$h\,(y)=h_o+\dfrac{(h_b-h_o)}{b}y$ at $y=y$,

and $h\,(y)=h$ at $h_o=h_b$

Using the assumed solution, equation (10) The relation between the thickness of the plate h(y) can be given by the following equation:

$$h(\eta) = h_o + (h_b - h_o)\eta \tag{31}$$

Appendix (B)

MATLAB code

Composite coefficients (function programs)

Appendix (B)

MATLAB Code

Composite Coefficients (function programs)

```
%***************************************************************************************************************************
function yEx = Ex(E1,E2,NU12,G12,theta)
%Ex This function returns the elastic modulus
% along the x-direction in the global
% coordinate system. It has five arguments:
% E1 - longitudinal elastic modulus
% E2 - transverse elastic modulus
% NU12 - Poisson's ratio
% G12 - shear modulus
% theta - fiber orientation angle
% The angle "theta" must be given in degrees.
% Ex is returned as a scalar
m = cos(theta*pi/180);
n = sin(theta*pi/180);
denom = m^4 + (E1/G12 - 2*NU12)*n*n*m*m + (E1/E2)*n^4;
yEx = E1/denom;
%***********************************************************THE end***********************************************************************
%********************************************CREATED BY WAEL A. AL-TABEY*****************************************************
%***************************************************************************************************************************
function yEy = Ey(E1,E2,NU21,G12,theta)
%Ey This function returns the elastic modulus
% along the y-direction in the global
% coordinate system. It has five arguments:
% E1 - longitudinal elastic modulus
% E2 - transverse elastic modulus
% NU21 - Poisson's ratio
% G12 - shear modulus
% theta - fiber orientation angle
% The angle "theta" must be given in degrees.
% Ey is returned as a scalar
m = cos(theta*pi/180);
n = sin(theta*pi/180);
denom = m^4 + (E2/G12 - 2*NU21)*n*n*m*m + (E2/E1)*n^4;
yEy = E2/denom;
%***********************************************************THE end***********************************************************************
%********************************************CREATED BY WAEL A. AL-TABEY*****************************************************
%***************************************************************************************************************************
function yGxy = Gxy(E1,E2,NU12,G12,theta)
%Gxy This function returns the shear modulus
% Gxy in the global
% coordinate system. It has five arguments:
% E1 - longitudinal elastic modulus
% E2 - transverse elastic modulus
% NU12 - Poisson's ratio
% G12 - shear modulus
% theta - fiber orientation angle
% The angle "theta" must be given in degrees.
% Gxy is returned as a scalar
m = cos(theta*pi/180);
n = sin(theta*pi/180);
denom = n^4 + m^4 + 2*(2*G12*(1 + 2*NU12)/E1 + 2*G12/E2 - 1)*n*n*m*m;
yGxy = G12/denom;
%***********************************************************THE end***********************************************************************
%********************************************CREATED BY WAEL A. AL-TABEY*****************************************************
%***************************************************************************************************************************
function yNUxy = NUxy(E1,E2,NU12,G12,theta)
```

```
%NUxy This function returns Poisson's ratio
% NUxy in the global
% coordinate system. It has five arguments:
% E1 - longitudinal elastic modulus
% E2 - transverse elastic modulus
% NU12 - Poisson's ratio
% G12 - shear modulus
% theta - fiber orientation angle
% The angle "theta" must be given in degrees.
% NUxy is returned as a scalar
m = cos(theta*pi/180);
n = sin(theta*pi/180);
denom = m^4 + (E1/G12 - 2*NU12)*n*n*m*m + (E1/E2)*n*n;
numer = NU12*(n^4 + m^4) - (1 + E1/E2 - E1/G12)*n*n*m*m;
yNUxy = numer/denom;
%**********************************************************THE end*******************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY****************************************************
%*********************************************************************************************************************************************
function yNUyx = NUyx(E1,E2,NU21,G12,theta)
%NUyx This function returns Poisson's ratio
% NUyx in the global
% coordinate system. It has five arguments:
% E1 - longitudinal elastic modulus
% E2 - transverse elastic modulus
% NU21 - Poisson's ratio
% G12 - shear modulus
% theta - fiber orientation angle
% The angle "theta" must be given in degrees.
% NUyx is returned as a scalar
m = cos(theta*pi/180);
n = sin(theta*pi/180);
denom = m^4 + (E2/G12 - 2*NU21)*n*n*m*m + (E2/E1)*n*n;
numer = NU21*(n^4 + m^4) - (1 + E2/E1 - E2/G12)*n*n*m*m;
yNUyx = numer/denom;
%**********************************************************THE end*******************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY****************************************************
%*********************************************************************************************************************************************
function y = OrthotropicCompliance(E1,E2,E3,NU12,NU23,NU13,G12,G23,G13)
%OrthotropicCompliance This function returns the compliance matrix
% for orthotropic materials. There are nine
% arguments representing the nine independent
% material constants. The size of the compliance
% matrix is 6 x 6.
y = [1/E1 -NU12/E1 -NU13/E1 0 0 0 ; -NU12/E1 1/E2 -NU23/E2 0 0 0 ;-NU13/E1 -...
NU23/E2 1/E3 0 0 0 ; 0 0 0 1/G23 0 0 ; 0 0 0 0 1/G13 0 ;0 0 0 0 0 1/G12];
%**********************************************************THE end*******************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY****************************************************
%*********************************************************************************************************************************************
function C = OrthotropicStiffness(E1,E2,E3,NU12,NU23,NU13,G12,G23,G13)
%OrthotropicStiffness This function returns the stiffness matrix
% for orthotropic materials. There are nine
% arguments representing the nine independent
% material constants. The size of the stiffness
% matrix is 6 x 6.
x = [1/E1 -NU12/E1 -NU13/E1 0 0 0 ; -NU12/E1 1/E2 -NU23/E2 0 0 0 ;-NU13/E1 -...
 NU23/E2 1/E3 0 0 0 ; 0 0 0 1/G23 0 0 ; 0 0 0 0 1/G13 0 ;0 0 0 0 0 1/G12];
C = inv(x);
%**********************************************************THE end*******************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY****************************************************
%*********************************************************************************************************************************************
function S = ReducedCompliance(E1,E2,NU12,G12)
```

http://dx.doi.org/10.5772/57384

```
%ReducedCompliance This function returns the reduced compliance
% matrix for fiber-reinforced materials.
% There are four arguments representing four
% material constants. The size of the reduced
% compliance matrix is 3 x 3.
S = [1/E1 -NU12/E1 0 ; -NU12/E1 1/E2 0 ; 0 0 1/G12];
%**********************************************************THE end*******************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY*******************************************************
%****************************************************************************************************************************************
function Q = ReducedStiffness(E1,E2,NU12,G12)
%ReducedStiffness This function returns the reduced stiffness
% matrix for fiber-reinforced materials.
% There are four arguments representing four
% material constants. The size of the reduced
% stiffness matrix is 3 x 3.
NU21 = NU12*E2/E1;
Q = [E1/(1-NU12*NU21) NU12*E2/(1-NU12*NU21) 0 ;NU12*E2/(1-NU12*NU21) E2/(1-...
NU12*NU21) 0 ; 0 0 G12];
%**********************************************************THE end*******************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY*******************************************************
%****************************************************************************************************************************************
function SBar = Sbar(S,theta)
%Sbar This function returns the transformed reduced
% compliance matrix "Sbar" given the reduced
% compliance matrix S and the orientation
% angle "theta".
% There are two arguments representing S and "theta"
% The size of the matrix is 3 x 3.
% The angle "theta" must be given in degrees.
R=[1 0 0;0 1 0;0 0 2];
Rinv=inv(R);
m = cos(theta*pi/180);
n = sin(theta*pi/180);
T = [m*m n*n 2*m*n ; n*n m*m -2*m*n ; -m*n m*n m*m-n*n];
Tinv = [m*m n*n -2*m*n ; n*n m*m 2*m*n ; m*n -m*n m*m-n*n];
SBar = Tinv*S*R*T*Rinv;
%**********************************************************THE end*******************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY*******************************************************
%****************************************************************************************************************************************
function QBar = Qbar(Q,theta)
%Qbar This function returns the transformed reduced
% stiffness matrix "Qbar" given the reduced
% stiffness matrix Q and the orientation
% angle "theta".
% There are two arguments representing Q and "theta"
% The size of the matrix is 3 x 3.
% The angle "theta" must be given in degrees.
R=[1 0 0;0 1 0;0 0 2];
Rinv=inv(R);
m = cos(theta*pi/180);
n = sin(theta*pi/180);
T = [m*m n*n 2*m*n ; n*n m*m -2*m*n ; -m*n m*n m*m-n*n];
Tinv = [m*m n*n -2*m*n ; n*n m*m 2*m*n ; m*n -m*n m*m-n*n];
QBar = Tinv*Q*R*T*Rinv;
%**********************************************************THE end*******************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY*******************************************************
%****************************************************************************************************************************************
function y = Amatrix(A,QBar,z1,z2)
%Amatrix This function returns the [A] matrix
% after the layer k with stiffness [Qbar]
% is assembled.
```

```
% A - [A] matrix after layer k
% is assembled.
% Qbar - [Qbar] matrix for layer k
% z1 - z(k-1) for layer k
% z2 - z(k) for layer k
for i = 1 : 3
for j = 1 : 3
A(i,j) = A(i,j) + QBar(i,j)*(z2-z1);
end
end
y = A;
%**********************************************************THE end***************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY*************************************************
%******************************************************************************************************************************
function y = Bmatrix(B,QBar,z1,z2)
%Bmatrix This function returns the [B] matrix
% after the layer k with stiffness [Qbar]
% is assembled.
% B - [B] matrix after layer k
% is assembled.
% Qbar - [Qbar] matrix for layer k
% z1 - z(k-1) for layer k
% z2 - z(k) for layer k
for i = 1 : 3
for j = 1 : 3
B(i,j) = B(i,j) + QBar(i,j)*(z2^2 -z1^2);
end
end
y = B;
%**********************************************************THE end***************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY*************************************************
%******************************************************************************************************************************
function y = Dmatrix(D,QBar,z1,z2)
%Dmatrix This function returns the [D] matrix
% after the layer k with stiffness [Qbar]
% is assembled.
% D - [D] matrix after layer k
% is assembled.
% Qbar - [Qbar] matrix for layer k
% z1 - z(k-1) for layer k
% z2 - z(k) for layer k
for i = 1 : 3
for j = 1 : 3
D(i,j) = D(i,j) + QBar(i,j)*(z2^3 -z1^3);
end
end
y = D;
%**********************************************************THE end***************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY*************************************************
%******************************************************************************************************************************
%******************************************************************************************************************************
%***********The Function MATLAB code for Laminate Analysis*****************
%**************calculate [A],[B]&[D] matrix for Laminate******************
%******************Using Orthotropic Qbar function***********************
%******************************************************************************************************************************
function [A,B,D]=ABDmatrix(E11,E22,NU12,G12,ho,n);
%******************************************************************************************************************************
% The reduced stiffness [Q] in GPa is calculated for this material
Q=ReducedStiffness(E11,E22,NU12,G12);
%******************************************************************************************************************************
A=zeros(3,3);
```

http://dx.doi.org/10.5772/57384

```
B=zeros(3,3);
D=zeros(3,3);
for i=1:n
z=-(ho/2):(ho/n):(ho/2);
theta(i)=input('The angle of the layer laminate (degree)=');
%The transformed reduced stiffnesses [¯Q] in GPa for the two layers are now calculated
QBar=Qbar(Q,theta(i));
%The [A] matrix is calculated
A=Amatrix(A,QBar,z(i),z(i+1));
%The [B] matrix is calculated
B=Bmatrix(B,QBar,z(i),z(i+1));
%The [D] matrix is calculated
D=Dmatrix(D,QBar,z(i),z(i+1));
end
A=A;
B=B/2;
D=D/3;
%************************************************************THE end*******************************************************************
%********************************************CREATED BY WAEL A. AL-TABEY****************************************************
%**************************************************************************************************************************************
%**************************************************************************************************************************************
%*******The Function MATLAB code for Laminate Coefficients***************
%*******calculate Epsy-Alfa-beta Coefficients for Laminate***************
%******************Using Orthotropic Qbar function***********************
%**************************************************************************************************************************************
function
[epsy1,epsy2,epsy3,epsy4,epsy5,epsy6,beta,alfa1,alfa2,alfa3]=EpsyAlfa(E11,E22,NU12,G12,ho,n,Nxbar,Nybar,Nxybar,a,)
%**************************************************************************************************************************************
%The [A],[B]&[D] matrix is calculated
[A,B,D]=ABDmatrix(E11,E22,NU12,G12,ho,n)
%**************************************************************************************************************************************
epsy1=D(1,1)/D(2,2)            %D11/D22
epsy2=(D(1,2)+(2*D(3,3)))/D(2,2)   %(D12+2D66)/D22
epsy3=D(1,3)/D(2,2)            %D16/D22
epsy4=D(2,3)/D(2,2)            %D26/D22
epsy5=D(1,2)/D(2,2)            %D12/D22
epsy6=(D(1,2)+(4*D(3,3)))/D(2,2)   %(D12+4D66)/D22
beta=a/b
alfa1=Nxbar/D(2,2)             %Nx/D22
alfa2=Nybar/D(2,2)             %Ny/D22
alfa3=Nxybar/D(2,2)             %Nxy/D22
%************************************************************THE end*******************************************************************
%********************************************CREATED BY WAEL A. AL-TABEY****************************************************
%**************************************************************************************************************************************
%**************************************************************************************************************************************
%*******The Function MATLAB code for Laminate Coefficients***************
%**calculate D03,Cf2,FR3,FR4,FT3,FT4 Coefficients for Boundary Conditions**
%**************************************************************************************************************************************
function [EI,FR1,FR2,FR3,FR4,FT1,FT2,FT3,FT4]=CoeffBC(E11,E22,NU12,G12,ho,n,a,b,R1,R2,R3,R4,T1,T2,T3,T4)
%**************************************************************************************************************************************
%The [A],[B]&[D] matrix is calculated
[A,B,D]=ABDmatrix(E11,E22,NU12,G12,ho,n);
%**************************************************************************************************************************************
EI=(1-(NU12^2))*D(1,1)              %(1-NU12^2)*D11
FR1=(R1*a)/EI                  %R1*a/EI
FR2=(R2*a)/EI                  %R2*a/EI
FR3=(R3*b)/D(2,2)                %R3*b/D22
FR4=(R4*b)/D(2,2)                %R4*b/D22
FT1=(T1*(a^3))/EI                %T1*a^3/EI
FT2=(T2*(a^3))/EI                %T2*a^3/EI
FT3=(T3*(b^3))/D(2,2)              %T3*b^3/D22
```

```
FT4=(T4*(b^3))/D(2,2)                %T4*b^3/D22
%***********************************************************THE end*******************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY*****************************************************
%********************************************************************************************************************************************
```

The main program (Call program)

```
%********************************************************************************************************************************************
%*********************Laminate Coefficients*******************************
%****epsy1,epsy2,epsy3,epsy4,beta,alfa1,alfa2,alfa3,Pbar,f1,f2,f3,f4*******
%******************lambda,D03,Cf2,FR3,FR4,FT3,FT4*************************
%********************************************************************************************************************************************
clc;
close all; clear all;
syms y ho hb b Eta hy
%********************************************************************************************************************************************
hy=input('The Thickness Equation of Layer Laminate as Function of y h(y)=');
%********************************************************************************************************************************************
 [t1,t2,t3,t4]=VTheckniss(y,ho,hb,b,Eta,hy);
%********************************************************************************************************************************************
E11=input('The Modulus of Elasticity E11(GPa)=');
E22=input('The Modulus of Elasticity E22(GPa)=');
NU12=input('The Poisson Coefficient NU12=');
G12=input('The Shear Modulus G12(GPa)=');
ho=input('The Inetial thickness of Layer Laminate at y = 0.0 ho (mm)=');
hb=input('The Final thickness of Layer Laminate at y = 0.0 ho (mm)=');
a=input('The Dimensions of Laminate at x Direction a(mm)=');
b=input('The Dimensions of Laminate at y Direction b(mm)=');
n=input('The Total Number of Laminate Layer n=');
Nxbar=input('The In-Plane Load in X Direction Nxbar(N)=');
Nybar=input('The In-Plane Load in Y Direction Nybar(N)=');
Nxybar=input('The In-Plane Load in XY Plane Nxybar(N)=');
R1=input('The Rotational Stiffness of support at x=0 R1 (N.m/rad)=');
R2=input('The Rotational Stiffness of support at x=a R2 (N.m/rad)=');
R3=input('The Rotational Stiffness of support at y=0 R3 (N.m/rad)=');
R4=input('The Rotational Stiffness of support at y=b R4 (N.m/rad)=');
T1=input('The Translation Stiffness of support at x=0 T1 (N/m)=');
T2=input('The Translation Stiffness of support at x=a T2 (N/m)=');
T3=input('The Translation Stiffness of support at y=0 T3 (N/m)=');
T4=input('The Translation Stiffness of support at y=b T4 (N/m)=');
%********************************************************************************************************************************************
t1=eval(t1)
t2=eval(t2)
t3=eval(t3)
t4=eval(t4)
%********************************************************************************************************************************************
[epsy1,epsy2,epsy3,epsy4,epsy5,epsy6,beta,alfa1,alfa2,alfa3]=EpsyAlfa(E11,E22,NU12,G12,ho,n,Nxbar,Nybar,Nxybar,a,);
%********************************************************************************************************************************************
 [EI,FR1,FR2,FR3,FR4,FT1,FT2,FT3,FT4]=CoeffBC(E11,E22,NU12,G12,ho,n,a,b,R1,R2,R3,R4,T1,T2,T3,T4);
%***********************************************************THE end*******************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY*****************************************************
```

http://dx.doi.org/10.5772/57384

The Beam Equations

```
%*******************************************************************************************************************************
echo off;
clc;
clear all;
echo on;
%*******************************************************************************************************************************
%*The MATLAB code for Calculate Constants Of The Equation of X of The Beam*
%************************A1, A2, A3, A4**********************************
%****Using Normalizatio of FR1 FR2 FT1 FT2 For Each Case Separately********
%***************And Estimated The Constant Fay in (C.22)******************
%*********************And Beam Equation X (C.39)*************************
%*******************************************************************************************************************************
echo off;
syms m A1 A2 A3 A4 FR1 FR2 FT1 FT2 zeta
a11=FR1;
a12= -m;
a13= FR1;
a14= m;
a21= m^3;
a22= FT1;
a23= -m^3;
a24= FT1;
a31= FR2*m*cosh(m)+(sinh(m))*m^2;
a32= FR2*m*sinh(m)+(cosh(m))*m^2;
a33=FR2*m*cos(m)-(sin(m))*m^2;
a34= -FR2*m*sin(m)-(cos(m))*m^2;
a41= FT2*sinh(m)-cosh(m)*m^3;
a42= FT2*cosh(m)-sinh(m)*m^3;
a43= FT2*sin(m)+cos(m)*m^3;
a44= FT2*cos(m)-sin(m)*m^3;
a=[a11,a12,a13,a14;a21,a22,a23,a24;a31,a32,a33,a34;a41,a42,a43,a44];%(C.9)
b=[A4;A3;A2;A1];
E=a*b;
A11=solve(E(1),A1);%(C.11)
A12=solve(E(2),A1);%(C.10)
A13=A11-A12;%(C.12)
A14=solve(A13,A2);%(C.14)
A15=subs(E(3),A11,A1);%(C.15)
A16=simplify(A15);%(C.16)
A17=subs(A16,A14,A2);%(C.17)
A18=simplify(A17);%(C.18)
A19=solve(A18,A3);%(C.21)
fay=subs(A19,A4,1)%(C.22)
A21=subs(E(4),A11,A1);%(C.23)
A22=simplify(A21);%(C.24)
A23=subs(A22,A14,A2);%(C.25)
A24=simplify(A23);%(C.26)
A25=solve(A24,A3);%(C.29)
fay1=subs(A25,A4,1)%(C.30)
MUEQ=fay-fay1%(C.31)
Ax1=solve(E(1),A2);%(C.32)
Ax2=subs(E(2),Ax1,A2);%(C.33)
Ax3=subs(Ax2,A19,A3);
Ax4=solve(Ax3,A4);%(C.35)*A4=**A1
Ax5=subs(A14,A19,A3);
Ax6=subs(Ax5,Ax4,A4);%(C.36)*A2=**A1
syms fay
Ax7=Ax6*fay;%(C.37)*A3=**A1
A=1;%A1 Assumption
```

```
B=subs(Ax6,A1,1);%A2
C=subs(Ax7,A1,1);%A3
D=subs(Ax4,A1,1);%A3
X=A*cos(m*zeta)+B*sin(m*zeta)+C*cosh(m*zeta)+D*sinh(m*zeta)
%************************************************************THE end**********************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY**************************************************
%********************************************************************************************************************************
echo off;
clc;
clear all;
echo on;
%********************************************************************************************************************************
%*The MATLAB code for Calculate Constants Of The Equation of X of The Beam*
%*************************A1, A2, A3, A4************************************
%****Using Normalizatio of FR1 FR2 FT1 FT2 For Each Case Separately********
%****************And Estimated The Constant Fay in (C.22)******************
%**********************And Beam Equation X (C.39)**************************
%********************************************************************************************************************************
echo off;
syms m A1 A2 A3 A4 FR1 FR2 FT1 FT2 zeta
BC=input('The Type of The Boundary Condition ( SS, CC, FF, CF, FR, FT, SC, TT, RR, RT, RS, TS, CR, CT, ES, CE, FE, EE)=','s');
if BC=='SS'
K1=0;
K2=0;
K3=1;
K4=1;
else if BC=='CC'
K1=1;
K2=1;
K3=1;
K4=1;
else if BC=='FF'
K1=0;
K2=0;
K3=0;
K4=0;
else if BC=='CF'
K1=1;
K2=0;
K3=1;
K4=0;
else if BC=='FR'
K1=0;
K2=1;
K3=0;
K4=1;
else if BC=='FT'
K1=0;
K2=0;
K3=0;
K4=1;
else if BC=='SC'
K1=0;
K2=1;
K3=1;
K4=1;
else if BC=='TT'
K1=0;
K2=0;
K3=1;
```

```
K4=1;
else if BC=='RR'
K1=1;
K2=1;
K3=1;
K4=1;
else if BC=='RT'
K1=1;
K2=0;
K3=1;
K4=1;
else if BC=='RS'
K1=1;
K2=0;
K3=1;
K4=1;
else if BC=='TS'
K1=0;
K2=0;
K3=1;
K4=1;
else if BC=='CR'
K1=1;
K2=1;
K3=1;
K4=1;
else if BC=='CT'
K1=1;
K2=0;
K3=1;
K4=1;
else if BC=='ES'%RTS
K1=1;
K2=0;
K3=1;
K4=1;
else if BC=='CE'%CRT
K1=1;
K2=1;
K3=1;
K4=1;
else if BC=='FE'%FRT
K1=0;
K2=1;
K3=0;
K4=1;
else if BC=='EE'%RTRT
K1=1;
K2=1;
K3=1;
K4=1;
end
if K1==1
a11=FR1;
a12= -m;
a13= FR1;
a14= m;
else
a11=1.0;
a12= -m/FR1;
a13= 1.0;
```

```
a14= m/FR1;
end
if K3==1
a21= m^3;
a22= FT1;
a23= -m^3;
a24= FT1;
else
a21= m^3/FT1;
a22= 1.0;
a23= -m^3/FT1;
a24= 1.0;
end
if K2==1
a31= FR2*m*cosh(m)+(sinh(m))*m^2;
a32= FR2*m*sinh(m)+(cosh(m))*m^2;
a33=FR2*m*cos(m)-(sin(m))*m^2;
a34= -FR2*m*sin(m)-(cos(m))*m^2;
else
a31= cosh(m)+(m/FR2)*sinh(m);
a32= sinh(m)+(m/FR2)*cosh(m);
a33= cos(m)-(m/FR2)*sin(m);
a34= -sin(m)-(m/FR2)*cos(m);
end
if K4==1
a41= FT2*sinh(m)-cosh(m)*m^3;
a42= FT2*cosh(m)-sinh(m)*m^3;
a43= FT2*sin(m)+cos(m)*m^3;
a44= FT2*cos(m)-sin(m)*m^3;
else
a41= sinh(m)-cosh(m)*m^3/FT2;
a42= cosh(m)-sinh(m)*m^3/FT2;
a43= sin(m)+cos(m)*m^3/FT2;
a44= cos(m)-sin(m)*m^3/FT2;
end
a=[a11,a12,a13,a14;a21,a22,a23,a24;a31,a32,a33,a34;a41,a42,a43,a44];%(C.9)
b=[A4;A3;A2;A1];
E=a*b;
A11=solve(E(1),A1);%(C.11)
A12=solve(E(2),A1);%(C.10)
A13=A11-A12;%(C.12)
A14=solve(A13,A2);%(C.14)
A15=subs(E(3),A11,A1);%(C.15)
A16=simplify(A15);%(C.16)
A17=subs(A16,A14,A2);%(C.17)
A18=simplify(A17);%(C.18)
A19=solve(A18,A3);%(C.21)
fay=subs(A19,A4,1)%(C.22)
A21=subs(E(4),A11,A1);%(C.23)
A22=simplify(A21);%(C.24)
A23=subs(A22,A14,A2);%(C.25)
A24=simplify(A23);%(C.26)
A25=solve(A24,A3);%(C.29)
fay1=subs(A25,A4,1)%(C.30)
MUEQ=fay-fay1%(C.31)
Ax1=solve(E(1),A2);%(C.32)
Ax2=subs(E(2),Ax1,A2);%(C.33)
Ax3=subs(Ax2,A19,A3);
Ax4=solve(Ax3,A4);%(C.35)*A4=**A1
Ax5=subs(A14,A19,A3);
Ax6=subs(Ax5,Ax4,A4);%(C.36)*A2=**A1
```

```
Ax7=Ax6*fay;%(C.37)*A3=**A1
A=1;%A1 Assumption
B=subs(Ax6,A1,1);%A2
C=subs(Ax7,A1,1);%A3
D=subs(Ax4,A1,1);%A3
X=A*cos(m*zeta)+B*sin(m*zeta)+C*cosh(m*zeta)+D*sinh(m*zeta)
%************************************************************THE end******************************************************************
%********************************************CREATED BY WAEL A. AL-TABEY**********************************************************
%**********************************************************************************************************************************
echo off;
clc;
clear all;
echo on;
%**********************************************************************************************************************************
%****The  MATLAB code for Calculate The Equation of Fay of The Beam********
%***************For All Cases Of Boundary Condations***********************
%******Using The Value of FR1 FR2 FT1 FT2 For Each Case Separately*********
%*************To Used The Equation of Fay in BeamSrip Programe*************
%*******************Using One Equation of Fay (C.34)************************
%**********************************************************************************************************************************
echo off;
syms FR1 FR2 FT1 FT2 m zeta
fay1=-(m^5*sinh(m)+sin(m)*m*FT1*FR1+m^4*FR2*cosh(m)+FR2*m^4*cos(m)+2*cos(m)*m^4*FR1-
sin(m)*m^5+FR2*cosh(m)*FR1*FT1+m*sinh(m)*FR1*FT1-FR2*cos(m)*FT1*FR1+2*FR2*sin(m)*FR1*m^3)/(-
2*sin(m)*m^2*FT1+m^5*cosh(m)-cos(m)*m^5+FR2*sinh(m)*FR1*FT1+m*cosh(m)*FR1*FT1-
FR2*sin(m)*m^4+2*FR2*m*cos(m)*FT1+m^4*FR2*sinh(m)+FR2*sin(m)*FR1*FT1+cos(m)*m*FR1*FT1);
BC=input('The Type of The Boundary Condition ( SS, CC, FF, CF, FR, FT, SC, TT, RR, RT, RS, TS, CR, CT, SE, CE,
FE)=','s');
if BC=='SS'
fay2=limit(fay1,FR1,0);
fay3=limit(fay2,FR2,0);
fay4=limit(fay3,FT1,inf);
fay=limit(fay4,FT2,inf)
else if BC=='CC'
fay2=limit(fay1,FR1,inf);
fay3=limit(fay2,FR2,inf);
fay4=limit(fay3,FT1,inf);
fay=limit(fay4,FT2,inf)
else if BC=='FF'
fay2=limit(fay1,FR1,0);
fay3=limit(fay2,FR2,0);
fay4=limit(fay3,FT1,0);
fay=limit(fay4,FT2,0)
else if BC=='FR'
fay2=limit(fay1,FR1,0);
fay3=limit(fay2,FT1,0);
fay=limit(fay3,FT2,inf)
else if BC=='FT'
fay2=limit(fay1,FR1,0);
fay3=limit(fay2,FR2,0);
fay=limit(fay3,FT1,0);
else if BC=='SC'
fay2=limit(fay1,FR1,0);
fay3=limit(fay2,FR2,inf);
fay4=limit(fay3,FT1,inf);
fay=limit(fay4,FT2,inf)
else if BC=='CF'
fay2=limit(fay1,FR1,inf);
fay3=limit(fay2,FR2,0);
fay4=limit(fay3,FT1,inf);
fay=limit(fay4,FT2,0)
```

```
else if BC=='TT'
fay2=limit(fay1,FR1,0);
fay=limit(fay2,FR2,0)
else if BC=='RR'
fay2=limit(fay1,FT1,inf);
fay=limit(fay2,FT2,inf)
else if BC=='RT'
fay2=limit(fay1,FR2,0);
fay=limit(fay2,FT1,inf)
else if BC=='RS'
fay2=limit(fay1,FR2,0);
fay3=limit(fay2,FT1,inf);
fay=limit(fay3,FT2,inf)
else if BC=='TS'
fay2=limit(fay1,FR1,0);
fay3=limit(fay2,FR2,0);
fay=limit(fay3,FT2,inf)
else if BC=='CR'
fay2=limit(fay1,FR1,inf);
fay3=limit(fay2,FT1,inf);
fay=limit(fay3,FT2,inf)
else if BC=='CT'
fay2=limit(fay1,FR1,inf);
fay3=limit(fay2,FR2,inf);
fay=limit(fay3,FT1,inf)
else if BC=='SE'
fay2=limit(fay1,FR1,0);
fay=limit(fay2,FT1,inf)
else if BC=='CE'
fay2=limit(fay1,FR1,inf);
fay=limit(fay2,FT1,inf)
else if BC=='FE'%FRT
fay2=limit(fay1,FR1,0);
fay=limit(fay2,FT1,0)
end
pretty(fay)
%************************************************************THE end**********************************************************************
%********************************************CREATED BY WAEL A. AL-TABEY****************************************************
%*****************************************************************************************************************************************
echo off;
clc;
clear all;
echo on;
%*****************************************************************************************************************************************
%*****The  MATLAB code for Calculate The Equation of MU of The Beam********
%**************For All Cases Of Boundary Condations**********************
%******Using The Value of FR1 FR2 FT1 FT2 For Each Case Separately*********
%******************Using One Equation of MU (C.31)***********************
%*****************************************************************************************************************************************
echo off;
syms FR1 FR2 FT1 FT2 m
eMU1=-(m^5*sinh(m)-
sin(m)*m^5+m^4*FR2*cosh(m)+FR2*m^4*cos(m)+2*cos(m)*m^4*FR1+FR2*cosh(m)*FR1*FT1+m*sinh(m)*FR1*FT1-
FR2*cos(m)*FT1*FR1+sin(m)*m*FT1*FR1+2*FR2*sin(m)*FR1*m^3)/(-FR2*sin(m)*m^4-
2*sin(m)*m^2*FT1+FR2*sinh(m)*FR1*FT1+m^4*FR2*sinh(m)+m^5*cosh(m)+2*FR2*m*cos(m)*FT1-
cos(m)*m^5+m*cosh(m)*FR1*FT1+FR2*sin(m)*FR1*FT1+cos(m)*m*FR1*FT1)+(m^7*cosh(m)-m^7*cos(m)-
m^4*FT2*sinh(m)-m^4*FT2*sin(m)-2*sin(m)*FR1*m^6-
FT2*sinh(m)*FR1*FT1+m^3*cosh(m)*FR1*FT1+FT2*sin(m)*FT1*FR1+m^3*cos(m)*FT1*FR1+2*FT2*cos(m)*FR1*m^3)/(m
^7*sinh(m)-FT2*cos(m)*m^4-2*m^4*cos(m)*FT1-FT2*cosh(m)*FR1*FT1-m^4*FT2*cosh(m)-
2*m*FT2*sin(m)*FT1+sin(m)*m^7+m^3*sinh(m)*FR1*FT1+FT2*cos(m)*FR1*FT1-sin(m)*m^3*FR1*FT1);
```

http://dx.doi.org/10.5772/57384

```
BC=input('The Type of The Boundary Condition ( SS, CC, FF, CF, FR, FT, SC, TT, RR, RT, RS, TS, CR, CT, SE, CE,
FE)=','s');
if BC=='SS'
eMU2=limit(eMU1,FR1,0);
eMU3=limit(eMU2,FR2,0);
eMU4=limit(eMU3,FT1,inf);
eMU=limit(eMU4,FT2,inf)
else if BC=='CC'
eMU2=limit(eMU1,FR1,inf);
eMU3=limit(eMU2,FR2,inf);
eMU4=limit(eMU3,FT1,inf);
eMU=limit(eMU4,FT2,inf)
else if BC=='FF'
eMU2=limit(eMU1,FR1,0);
eMU3=limit(eMU2,FR2,0);
eMU4=limit(eMU3,FT1,0);
eMU=limit(eMU4,FT2,0)
else if BC=='FR'
eMU2=limit(eMU1,FR1,0);
eMU3=limit(eMU2,FT1,0);
eMU=limit(eMU3,FT2,inf)
else if BC=='FT'
eMU2=limit(eMU1,FR1,0);
eMU3=limit(eMU2,FR2,0);
eMU=limit(eMU3,FT1,0);
else if BC=='SC'
eMU2=limit(eMU1,FR1,0);
eMU3=limit(eMU2,FR2,inf);
eMU4=limit(eMU3,FT1,inf);
eMU=limit(eMU4,FT2,inf)
else if BC=='CF'
eMU2=limit(eMU1,FR1,inf);
eMU3=limit(eMU2,FR2,0);
eMU4=limit(eMU3,FT1,inf);
eMU=limit(eMU4,FT2,0)
else if BC=='TT'
eMU2=limit(eMU1,FR1,0);
eMU=limit(eMU2,FR2,0)
else if BC=='RR'
eMU2=limit(eMU1,FT1,inf);
eMU=limit(eMU2,FT2,inf)
else if BC=='RT'
eMU2=limit(eMU1,FR2,0);
eMU=limit(eMU2,FT1,inf)
else if BC=='RS'
eMU2=limit(eMU1,FR2,0);
eMU3=limit(eMU2,FT1,inf);
eMU=limit(eMU3,FT2,inf)
else if BC=='TS'
eMU2=limit(eMU1,FR1,0);
eMU3=limit(eMU2,FR2,0);
eMU=limit(eMU3,FT2,inf)
else if BC=='CR'
eMU2=limit(eMU1,FR1,inf);
eMU3=limit(eMU2,FT1,inf);
eMU=limit(eMU3,FT2,inf)
else if BC=='CT'
eMU2=limit(eMU1,FR1,inf);
eMU3=limit(eMU2,FR2,inf);
eMU=limit(eMU3,FT1,inf)
else if BC=='SE'
```

```
eMU2=limit(eMU1,FR1,0);
eMU=limit(eMU2,FT1,inf)
else if BC=='CE'
eMU2=limit(eMU1,FR1,inf);
eMU=limit(eMU2,FT1,inf)
else if BC=='FE'%FRT
eMU2=limit(eMU1,FR1,0);
eMU=limit(eMU2,FT1,0)
end
pretty(eMU)
%************************************************************THE end**********************************************************************
%*********************************************CREATED BY WAEL A. AL-TABEY*****************************************************
%*******************************************************************************************************************************************
echo off;
clc;
clear all;
echo on;
%*******************************************************************************************************************************************
%****The  MATLAB code for Calculate The Equation of X of The Beam********
%***************For All Cases Of Boundary Condations***********************
%******Using The Value of FR1 FR2 FT1 FT2 For Each Case Separately*********
%*************To Used The Equation of Fay in BeamSrip Programe*************
%*******************Using One Equation of X (C.39)************************
%*******************************************************************************************************************************************
echo off;
syms FR1 FR2 FT1 FT2 m zeta fay
x1=sin(m*zeta)-((2*FR1*m*fay+FR1*FT1*(m^4)-1)/(fay-FR1*FT1*(m^4)*fay+2*FT1*(m^3)))*cos(m*zeta)-
((1+FR1*FT1*(m^4))/(fay-FR1*FT1*(m^4)*fay+2*FT1*(m^3)))*(fay*sinh(m*zeta)+cosh(m*zeta));
BC=input('The Type of The Boundary Condition ( SS, CC, FF, CF, FR, FT, SC, TT, RR, RT, RS, TS, CR, CT, SE, CE,
FE)=','s');
if BC=='SS'
x2=limit(x1,FR1,0);
x3=limit(x2,FR2,0);
x4=limit(x3,FT1,inf);
x=limit(x4,FT2,inf)
else if BC=='CC'
x2=limit(x1,FR1,inf);
x3=limit(x2,FR2,inf);
x4=limit(x3,FT1,inf);
x=limit(x4,FT2,inf)
else if BC=='FF'
x2=limit(x1,FR1,0);
x3=limit(x2,FR2,0);
x4=limit(x3,FT1,0);
x=limit(x4,FT2,0)
else if BC=='FR'
x2=limit(x1,FR1,0);
x3=limit(x2,FT1,0);
x=limit(x3,FT2,inf)
else if BC=='FT'
x2=limit(x1,FR1,0);
x3=limit(x2,FR2,0);
x=limit(x3,FT1,0);
else if BC=='SC'
x2=limit(x1,FR1,0);
x3=limit(x2,FR2,inf);
x4=limit(x3,FT1,inf);
x=limit(x4,FT2,inf)
else if BC=='CF'
x2=limit(x1,FR1,inf);
x3=limit(x2,FR2,0);
```

```
x4=limit(x3,FT1,inf);
x=limit(x4,FT2,0)
else if BC=='TT'
x2=limit(x1,FR1,0);
x=limit(x2,FR2,0)
else if BC=='RR'
x2=limit(x1,FT1,inf);
x=limit(x2,FT2,inf)
else if BC=='RT'
x2=limit(x1,FR2,0);
x=limit(x2,FT1,inf)
else if BC=='RS'
x2=limit(x1,FR2,0);
x3=limit(x2,FT1,inf);
x=limit(x3,FT2,inf)
else if BC=='TS'
x2=limit(x1,FR1,0);
x3=limit(x2,FR2,0);
x=limit(x3,FT2,inf)
else if BC=='CR'
x2=limit(x1,FR1,inf);
x3=limit(x2,FT1,inf);
x=limit(x3,FT2,inf)
else if BC=='CT'
x2=limit(x1,FR1,inf);
x3=limit(x2,FR2,inf);
x=limit(x3,FT1,inf)
else if BC=='SE'
x2=limit(x1,FR1,0);
x=limit(x2,FT1,inf)
else if BC=='CE'
x2=limit(x1,FR1,inf);
x=limit(x2,FT1,inf)
else if BC=='FE'%FRT
x2=limit(x1,FR1,0);
x=limit(x2,FT1,0)
end
pretty(x)
%****************************************************************THE end*********************************************************************
%**********************************************CREATED BY WAEL A. AL-TABEY**************************************************
%**************************************************************************************************************************************
echo off;
clc;
clear all;
echo on;
%**************************************************************************************************************************************
%*************The  MATLAB code for Plot The Beam Mode Shape*****************
%******************For All Cases Of Boundary Condations*********************
%**************************************************************************************************************************************
%FOR SS (FR1=0,   FR2=0,   FT1=inf,   FT2=inf)**
%**************************************************************************************************************************************
%FOR CC (FR1=inf,   FR2=inf,   FT1=inf,   FT2=inf)
%**************************************************************************************************************************************
%FOR FF (FR1=0,   FR2=0,   FT1=0,   FT2=0)**
%**************************************************************************************************************************************
%FOR CF (FR1=inf,   FR2=0,   FT1=inf,   FT2=0)**
%**************************************************************************************************************************************
%FOR FR (FR1=0,   FR2=FR,   FT1=0,   FT2=inf)**
%**************************************************************************************************************************************
%FOR FT (FR1=0,   FR2=0,   FT1=0,   FT2=FT)**
%**************************************************************************************************************************************
```

```
%FOR SC (FR1=0,  FR2=inf,  FT1=inf,  FT2=inf)**
%****************************************************************************************************
%FOR TT (FR1=0,  FR2=0,  FT1=FT2=FT)**
%****************************************************************************************************
%FOR RR (FR1=FR2=FR,  FT1=inf,  FT2=inf)**
%****************************************************************************************************
%FOR RT (FR1=FR,  FR2=0,  FT1=inf,  FT2=FT)**
%****************************************************************************************************
%FOR RS (FR1=FR,  FR2=0,  FT1=inf,  FT2=inf)**
%****************************************************************************************************
%FOR TS (FR1=0,  FR2=0,  FT1=FT,  FT2=inf)**
%****************************************************************************************************
%FOR CR (FR1=inf,  FR2=FR,  FT1=inf,  FT2=inf)
%****************************************************************************************************
%FOR CT (FR1=inf,  FR2=0,  FT1=inf,  FT2=FT)
%****************************************************************************************************
%FOR ES (FR1=FR,  FR2=0,  FT1=FT,  FT2=inf)**
%****************************************************************************************************
%FOR CE (FR1=inf,  FR2=FR,  FT1=inf,  FT2=FT)**
%****************************************************************************************************
%FOR FE (FR1=0,  FR2=FR,  FT1=0,  FT2=FT)**
%****************************************************************************************************
%FOR EE (FR1=FR2=FR,  FT1=FT2=FT)**
%****************************************************************************************************
echo off;
format long
syms m
BC=input('The Type of The Boundary Condition ( SS, CC, FF, CF, FR, FT, SC, TT, RR, RT, RS, TS, CR, CT, ES, CE, FE,
EE)=','s');
if BC=='SS'
FR1=0.00000000001;
FR2=0.00000000001;
FT1=10000000000;
FT2=10000000000;
else if BC=='CC'
FR1=10000000000;
FR2=10000000000;
FT1=10000000000;
FT2=10000000000;
else if BC=='FF'
FR1=0.00000000001;
FR2=0.00000000001;
FT1=0.00000000001;
FT2=0.00000000001;
else if BC=='CF'
FR1=10000000000;
FR2=0.00000000001;
FT1=10000000000;
FT2=0.00000000001;
else if BC=='FR'
FR1=0.00000000001;
FR2=input('The Rotational Stiffness of support at x=a R2 (N.m/rad)=');
FT1=0.00000000001;
FT2=10000000000;
else if BC=='FT'
FR1=0.00000000001;
FR2=0.00000000001;
FT1=0.00000000001;
FT2=input('The Translation Stiffness of support at x=a T2 (N/m)=');
else if BC=='SC'
FR1=0.00000000001;
```

http://dx.doi.org/10.5772/57384

```
FR2=10000000000;
FT1=10000000000;
FT2=10000000000;
else if BC=='TT'
FR1=0.00000000001;
FR2=0.00000000001;
FT1=input('The Translation Stiffness of support at x=0 T1(N/m)=');
FT2=input('The Translation Stiffness of support at x=a T2(N/m)=');
else if BC=='RR'
FR1=input('The Rotational Stiffness of support at x=0 R1 (N.m/rad)=');
FR2=input('The Rotational Stiffness of support at x=a R2 (N.m/rad)=');
FT1=10000000000;
FT2=10000000000;
else if BC=='RT'
FR1=input('The Rotational Stiffness of support at x=0 R1 (N.m/rad)=');
FR2=0.00000000001;
FT1=10000000000;
FT2=input('The Translation Stiffness of support at x=a T2(N/m)=');
else if BC=='RS'
FR1=input('The Rotational Stiffness of support at x=0 R1 (N.m/rad)=');
FR2=0.00000000001;
FT1=10000000000;
FT2=10000000000;
else if BC=='TS'
FR1=0.00000000001;
FR2=0.00000000001;
FT1=input('The Translation Stiffness of support at x=0 T1(N/m)=');
FT2=10000000000;
else if BC=='CR'
FR1=10000000000;
FR2=input('The Rotational Stiffness of support at x=a R2 (N.m/rad)=');
FT1=10000000000;
FT2=10000000000;
else if BC=='CT'
FR1=10000000000;
FR2=0.00000000001;
FT1=10000000000;
FT2=input('The Translation Stiffness of support at x=a T2(N/m)=');
else if BC=='ES'%RTS
FR1=input('The Rotational Stiffness of support at x=0 R1(N.m/rad)=');
FR2=0.00000000001;
FT1=input('The Translation Stiffness of support at x=0 T1(N/m)=');
FT2=10000000000;
else if BC=='CE'%CRT
FR1=10000000000;
FR2=input('The Rotational Stiffness of support at x=a R2 (N.m/rad)=');
FT1=10000000000;
FT2=input('The Translation Stiffness of support at x=a T2(N/m)=');
else if BC=='FE'%FRT
FR1=0.00000000001;
FR2=input('The Rotational Stiffness of support at x=a R2 (N.m/rad)=');
FT1=0.00000000001;
FT2=input('The Translation Stiffness of support at x=a T2(N/m)=');
else if BC=='EE'%RTRT
FR1=input('The Rotational Stiffness of support at x=0 R1 (N.m/rad)=');
FR2=input('The Rotational Stiffness of support at x=a R2 (N.m/rad)=');
FT1=input('The Translation Stiffness of support at x=0 T1(N/m)=');
FT2=input('The Translation Stiffness of support at x=a T2(N/m)=');
end
syms m
zeta=1;
```

```
syms zeta
N=input('The Number of Plate Strips N=');
R1=input('The Rotational Stiffness of support at x=0 R1 (N.m/rad)=');
R2=input('The Rotational Stiffness of support at x=a R2 (N.m/rad)=');
T1=input('The Translation Stiffness of support at x=0 T1 (N/m)=');
T2=input('The Translation Stiffness of support at x=a T2 (N/m)=');
NU12=input('The Poisson Coefficient NU12=');
a=input('The Dimensions of Laminate at x Direction a(mm)=');
D=input('enter D(1,1)=');
EI=(1-(NU12^2))*D;
FR1=(R1*a)/EI;                %R1*a/EI
FR2=(R2*a)/EI;                %R2*a/EI
FT1=(T1*(a^3))/EI;             %T1*a^3/EI
FT2=(T2*(a^3))/EI;              %T2*a^3/EI
for i=1:N
for j=1:N
m(i)=i*pi;
m(j)=j*pi;
fay(i)=((((2*FR1*FR2*(m(i).^4))-(m(i).^6)+(FR1*FT1*(m(i).^2))).*sin(m(i)))+(((2*FR1*(m(i).^5))+(FR2*(m(i).^5))-
(FR1*FR2*FT1*m(i))).*cos(m(i)))+(((m(i).^6)+(FR1*FT1*(m(i).^2))).*sinh(m(i)))+((((m(i).^5)*FR2)+(FR1*FR2*FT1*m(i))).*co
sh(m(i)))))./((((FR1*FR2*FT1*m(i))-(2*FT1*(m(i).^3))-
(FR2*(m(i).^5))).*sin(m(i)))+((((FR1*FT1*(m(i).^2))+(2*FR2*FT1*(m(i).^2))-
(m(i).^6))).*cos(m(i)))+(((FR2*(m(i).^5))+(FR1*FR2*FT1)).*sinh(m(i)))+(((m(i).^6)+(FR1*FT1*(m(i).^2))).*cosh(m(i))))
fay(j)=((((2*FR1*FR2*(m(j).^4))-(m(j).^6)+(FR1*FT1*(m(j).^2))).*sin(m(j)))+(((2*FR1*(m(j).^5))+(FR2*(m(j).^5))-
(FR1*FR2*FT1*m(j))).*cos(m(j)))+(((m(j).^6)+(FR1*FT1*(m(j).^2))).*sinh(m(j)))+((((m(j).^5)*FR2)+(FR1*FR2*FT1*m(j))).*cos
h(m(j)))))./((((FR1*FR2*FT1*m(j))-(2*FT1*(m(j).^3))-
(FR2*(m(j).^5))).*sin(m(j)))+((((FR1*FT1*(m(j).^2))+(2*FR2*FT1*(m(j).^2))-
(m(i).^6))).*cos(m(j)))+(((FR2*(m(j).^5))+(FR1*FR2*FT1)).*sinh(m(j)))+(((m(j).^6)+(FR1*FT1*(m(j).^2))).*cosh(m(j))))
x(i)=cos(m(i)*zeta)+((((-2*m(i).*FT1*fay(i)).*((m(i).^4)-(FT1*FR1)))-(((m(i).^4)-(FT1*FR1))^2))./((((m(i).^8)-
((FT1*FR1)^2)).*fay(i))-((2*(m(i)^3)*FR1).*((m(i)^4)+(FR1*FT1))))).*sin(m(i)*zeta)-((((m(i).^4)-
(FT1*FR1))).*fay(i))./((((m(i).^4)-(FT1*FR1)))-((2*(m(i)^3)*FR1))).*(sinh(m(i)*zeta)+(fay(i).*cosh(m(i)*zeta)))
dx(i)=diff(x(i),zeta)
ddx(i)=diff(dx(i),zeta)
dddx(i)=diff(ddx(i),zeta)
ddddx(i)=diff(dddx(i),zeta)
x(j)=cos(m(j)*zeta)+((((-2*m(j).*FT1*fay(j)).*((m(j).^4)-(FT1*FR1)))-(((m(j).^4)-(FT1*FR1))^2))./((((m(j).^8)-
((FT1*FR1)^2)).*fay(j))-((2*(m(j)^3)*FR1).*((m(j)^4)+(FR1*FT1))))).*sin(m(j)*zeta)-((((m(j).^4)-
(FT1*FR1))).*fay(j))./((((m(j).^4)-(FT1*FR1)))-((2*(m(j)^3)*FR1))).*(sinh(m(j)*zeta)+(fay(j).*cosh(m(j)*zeta)))
f1(i,j)=x(i)*x(j)
f2(i,j)=dx(i)*x(j)
f3(i,j)=ddx(i)*x(j)
f4(i,j)=dddx(i)*x(j)
f5(i,j)=ddddx(i)*x(j)
f6(i,j)=x(j)
a=int(f1,zeta,0,1)
b=int(f2,zeta,0,1)
c=int(f3,zeta,0,1)
d=int(f4,zeta,0,1)
e=int(f5,zeta,0,1)
p=int(f6,zeta,0,1)
end
end
a=eval(a)
b=eval(b)
c=eval(c)
d=eval(d)
e=eval(e)
p=eval(p)
%*************************************************************THE end**************************************************************
%*******************************************CREATED BY WAEL A. AL-TABEY***************************************************
%*************************************************************************************************************************************
```

http://dx.doi.org/10.5772/57384

```
%**********************************************************************************************************************
%********The programe to calculate the initial value **********************
%***********************BOUNDARY CONDITIONS********************************
%**********************************************************************************************************************
clc;
clear all;
%**********************************************************************************************************************
for i=1:m
   mi=4*i-2
   j=i
   yyi(mi,j)=1
end
%**********************************************************************************************************************
for i=1:m
   for j=1:m
      j=i
       mi=4*i
      yyi(mi,j)=-a24*(CK(i,j)/AK(i,j))
   end
end
%**********************************************************************************************************************
%********************New for B.C.against rotation**************************
%**********************************************************************************************************************
for i=1:m
   j=i
   mi=4*i-1
   yyi(mi,j)=1/R3
end
%**********************************************************************************************************************
for i=1:m
   jj=i+m
   mii=4*i-3
   yyi(mii,jj)=1
end
%**********************************************************************************************************************
for i=1:m
   for j=1:m
      j=i
      mii=4*i-1
      jj=j+m
      yyi(mii,jj)=-C13*(CK(i,j)/AK(i,j))
   end
end
%**********************************************************************************************************************
for i=1:m
   mii=4*i
   jj=i+m
   yyi(mii,jj)=-1/T3
end
%**********************************************************THE end*******************************************************
%********************************************CREATED BY WAEL A. AL-TABEY**************************************************
%**********************************************************************************************************************
%*******The programe to calculate the eigenvalue and eigenvvector**********
%**********************************************************************************************************************
clc;
clear all;
%***************************The eigenvalue*******************************
syms k1 k3
Ak=[0 1 0 0;0 0 1 0;0 0 0 1;-k1 0 -k3 0]
Lampda=eig(Ak)
%**********************************************************************************************************************
```

The Plate Mode Shape

```
%*****************************************************************************************************************************
%********************** The Plate Mode Shape ***************************
%*****************************************************************************************************************************
echo on
clc;
clear all;
pause % Strike any key to continue.
clc
%*********************Problem definition***********************************
g='squareg'; % The unit square
b='squareb3'; % 0 on the left and right boundaries and
%**********0 normal derivative on the top and bottom boundaries************
c=1;
a=0;
f=0;
d=1;
%*******************************Mesh*************************************
[p,e,t]=initmesh('squareg');
pause % Strike any key to continue.
clc
%***************************************************************************
x=p(1,:)';
y=p(2,:)';
%***************************************************************************
u0=atan(cos(pi/2*x));
ut0=3*sin(pi*x).*exp(sin(pi/2*y));
pause % Strike any key to continue.
clc
%*******We want the solution at 31 points in time between 0 and 5**********
n=31;
tlist=linspace(0,5,n);

%*************************Solve PDE problem of Plate***********************
uu=hyperbolic(u0,ut0,tlist,b,p,e,t,c,a,f,d);
pause % Strike any key to continue.
clc
%*****To speed up the plotting, we interpolate to a rectangular grid*******
delta=-1:0.1:1;
[uxy,tn,a2,a3]=tri2grid(p,t,uu(:,1),delta,delta);
gp=[tn;a2;a3];
%***************************Make the animation*****************************
newplot;
M=moviein(n);
umax=max(max(uu));
umin=min(min(uu));
for i=1:n,
 if rem(i,10)==0,
  fprintf('%d ',i);
 end
 pdeplot(p,e,t,'xydata',uu(:,i),'zdata',uu(:,i),'zstyle','continuous','mesh','on','xygrid','on','gridparam',gp,'colorbar','on');
 grid on
 colormap
 axis([-1 1 -1 1 umin umax]); caxis([umin umax]);
 title('Plate mode shape')
 xlabel('X')
 ylabel('Y')
 zlabel('Z')
 M(:,i)=getframe;
 if i==n;
```

http://dx.doi.org/10.5772/57384

```
   fprintf('done\n');
  end
end
%****************************Show movie*********************************
nfps=5;
movie(M,10,nfps);
pause % Strike any key to end.
echo off
%*************************************************************THE end*********************************************************************
%***********************************************CREATED BY WAEL A. AL-TABEY*************************************************
%*********************************************************************************************************************************
```

Author details

Wael A. Al-Tabey

Department of Mechanical Engineering Faculty of Engineering, Alexandria University, Alexandria, Egypt

References

[1] Utku M., Citipitioglu E. and Inceleme I, 2000, Circular plate on elastic foundation modeled with annular plates, J. Computinal Structures, 74, 78-365.

[2] Chadrashekhara K. and Antony J., 1997, Elastic analysis of an annular slab-soil interaction problem using hybrid method. Compt. Geotech, 76, 20-161.

[3] Ng SF, Araar Y, 1989, Free vibration and buckling analysis of clamped rectangular plates of variable thickness by Galerkin Method. J. Sound Vibration, 135(2):263-74.

[4] Ungbhakorn V and Singhatanadgid P, 2006, Buckling analysis of symmetrically laminated composite plates by the extended Kantorovich method, J. composite Structures, 73, 120-123.

[5] Setoodeh A.R, Karami G, 2003, A solution for the vibration and buckling of composite laminates with elastically restrained edges, J. composite Structures, 60, 245-253.

[6] Laura P. and Gutierrez R., 1985, Vibrating non uniform plates on elastic foundation. J. Mechanical Engineering. ASCE, 96 111-1185.

[7] Ashour A.S, 2004, vibration of variable thickness plates with edges elastically restrained against rotational and translation, Thin-Walled Structures, 42, 1-24.

[8] Grossi R.O, Nallim L.G, 2008, On the existence of weak solutions of anisotropic generally restrained plates, J. Applied Mathematical Modelling, 32, 2254-2273.

[9] LU C.F, Lee Y.Y, Lim C.W and Chen W.Q, 2006, free vibration of long-span contagious rectangular Kirchhoff plates with internal rigid line supports, J. Sound and Vibration, 297, 351-364.

[10] Chopra I. 1974, Vibration of stepped thickness plates. International Journal of Mechanical Science; 16:337–44.

[11] Cortinez VH, Laura PAA. 1990, Analysis of vibrating rectangular plates of discontinuously varying thickness by means of the Kantorovich extended method. J. of Sound and Vibration; 137(3):457–61.

[12] Bambill DV, Laura PAA, Bergmann A, Carnicer R. 1991, Fundamental frequency of transverse vibration of symmetrically stepped simply supported rectangular plates. J. of Sound and Vibration; 150(1):167–9.

[13] Laura PAA, Gutierrez RH. 1985, Transverse vibrations of rectangular plates on inhomogeneous foundations, Part I: Rayleigh-Ritz method. J. of Sound and Vibration; 101(3):307–15.

[14] Harik IE, Andrade MG. 1989, Stability of plates with step variation in thickness. J. Computers and Structures; 33(1):257–63.

[15] Cheung Y.K and Zhou D 2001, Vibration analysis of symmetrically laminated rectangular plates with intermediate line-supports, J. Thin-walled Structures, 79, 33-41.